mi computer

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

TOMO 7

Editorial Delta, S.A.


mi COMPUTER

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, F. Martín

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

ISBN: 84-7598-067-2 (tomo 7) 84-85822-82-X (obra completa)
Depósito legal: B. 52/1984
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona)
Impreso en España - Printed in Spain - 1985

mi COMPUTER

VOLUMEN 7

Editorial Delta, S.A.

Notas clave

El Music Maker, de Commodore, es un paquete ingeniosamente simple basado en un teclado de piano que se engancha encima del ordenador

Ciertas facilidades de los ordenadores personales invitan al desarrollo de periféricos que las amplíen para aprovechar al máximo su potencial. Esto es especialmente cierto en el caso de las capacidades para gráficos y sonido, para las cuales se ha creado una enorme gama de productos accesorios. Tales periféricos son particularmente necesarios en el Commodore, que carece incluso de las sencillas instrucciones de sonido de que disponen la mayoría de los otros micros. Para esta máquina se han puesto a la venta numerosos paquetes para componer música, pero, a pesar de sus méritos, las melodías aún se deben ejecutar en el teclado tipo máquina de escribir y no en el teclado de piano con el cual están familiarizados los músicos. Sin embargo, con la introducción del Music Maker, Commodore ha producido un paquete que proporciona acceso directo al sumamente veloz chip SID *(Sound Interface Device)* del micro y una cubierta de teclado que permite que el usuario utilice el familiar teclado blanco y negro del piano.

La cubierta, construida en plástico sólido, se encaja con comodidad sobre el cuerpo del teclado del ordenador, más o menos sobre las teclas de función y de máquina de escribir. El teclado Music Maker, que abarca dos octavas, es de un plástico más blando y cada tecla está fijada a la cubierta de forma individual. En la cara inferior de las teclas hay unas proyecciones en forma de dientes que hacen que al pulsar la tecla se haga presión sobre una tecla de máquina de escribir específica de las de abajo, que ha sido programada para producir una cierta nota.

Tratándose de un método tan económico y simple, el sistema funciona sorprendentemente bien. El plástico sólido de la cubierta mantiene con firmeza en su sitio las teclas del piano, y aun cuando el teclado se toque con rapidez de un extremo a otro, es muy raro que se pierdan notas.

El software que viene con el paquete se suministra en cassette o en disco. El programa es activado por menú, principalmente mediante las ocho teclas de función. Éstas permiten al usuario alterar el sonido (o número de sonidos) producido con el teclado de piano, cambiando ya sea la forma de la onda de sonido o bien el tono. Asimismo se proporcionan facilidades para utilizar un ritmo de percusión o bajo a modo de acompañamiento, y un secuenciador permite programar y reproducir melodías. Los sonidos y las frases programadas se pueden guardar (SAVE) o cargar (LOAD) en disco o cassette.

Creador de melodías

El paquete Melody Maker, para el Commodore 64, se compone de una cubierta de teclado de piano y el software para controlarla. La cubierta, de dos octavas, se instala en la estructura plástica del ordenador, de modo que las teclas de piano hacen contacto con las teclas del micro que están programadas para producir las correspondientes notas musicales. La cubierta deja a la vista las teclas de función, dado que éstas son necesarias para programar el sonido. El paquete produce un sonido de gran calidad

Ian McKinnell

Es posible programar ocho voces distintas para producir un sonido diferente. Utilizando la opción F6 (modificar voz), el usuario puede seleccionar la voz a alterar. Luego aparece en la pantalla una serie de opciones, comenzando por *attack* (ataque), *decay* (decaimiento), *sustain* (sostenimiento) y *release* (liberación), eligiéndose un número comprendido entre 0 y 15 para cada uno de estos parámetros. Se le pregunta luego al programador si se han de implementar filtros. (Éstos suprimen sonidos por arriba o por debajo de frecuencias especificadas.) Si uno responde que sí, aparece otra serie de preguntas para determinar su frecuencia y nivel.

El secuenciador es un importante componente de la música sintetizada que, curiosamente, se subvalora en muchos paquetes de música para ordenador. Programado en dos partes, ejecuta una frase repetida. Primero, el programador entra las notas de la frase a través del teclado de piano; después se graba la longitud de cada nota (y, por tanto, el ritmo de la melodía) utilizando la tecla F5.

Los tres ritmos de percusión que se ofrecen son

MUSIC MAKER

SOFTWARE

El software que se suministra con el paquete viene en formato de disco o de cassette

DOCUMENTACION

Se entrega con dos manuales. La guía para el usuario explica lo que es preciso saber para iniciarse, si bien carece de la clase de información detallada que necesitarían los usuarios más experimentados. Un segundo manual, *Start playing keyboard*, ofrece las partituras de 28 melodías populares

VENTAJAS

Por su precio, es un paquete muy valioso, que permite utilizar el Commodore 64 como un verdadero instrumento

DESVENTAJAS

La gama de opciones del paquete es notablemente restringida, en especial con respecto a los ritmos preprogramados de batería y bajo

limitados pero adecuados. Los ritmos de bajo siguen el mismo patrón que la percusión y el programador dispone de tres opciones que le permiten conectar o desconectar el bajo y variar la altura (*pitch*). Según el ritmo que se esté ejecutando, éste bajará la altura de la línea del bajo en un quinto de octava o bien en una octava completa. La velocidad del ritmo se puede alterar pulsando las teclas del cursor: Down (abajo) retarda el compás y Right (derecha) lo acelera.

No obstante, muchas de las opciones no se pueden utilizar juntas. Aunque el bajo y la percusión se pueden ejecutar de forma simultánea, tocando el usuario una melodía en el teclado de arriba, esto sólo se puede realizar con el teclado en modalidad monofónica. En consecuencia, es imposible tocar acordes con ritmo de fondo.

Ello también ocurre con la opción de secuenciador. Muchos grupos *pop* actuales omiten la sección rítmica convencional de bajo y batería y actúan, en cambio, con el apoyo de un secuenciador. En el Music Maker, sin embargo, no existe una facilidad para hacer eso, ni siquiera para reproducir el secuenciador con la sección de ritmo proporcionada. En la actualidad esta limitación se debe al hardware. Dado que el chip SID posee sólo tres voces, uno no puede esperar un ritmo de bajo y percusión y sonido polifónico al mismo tiempo. Con la sección de ritmo no funcionará siquiera el *glissando* (una opción que desliza el sonido de una nota a otra mientras se pulsa la tecla junto con la barra espaciadora), si bien es probable que este problema sea consecuencia del hecho de que el sonido se genera de forma digital. Mantener el bajo y la batería, además de producir un cambio suficientemente pequeño en la altura para producir una ligadura, parece estar más allá de las capacidades del procesador.

Más difícil de comprender es por qué quienes escribieron el software decidieron no permitir que el propio usuario programara sus ritmos de bajo y percusión en la máquina. Las técnicas serían las mismas que las utilizadas en el secuenciador, y una sección de ritmo programable le hubiera añadido al paquete una gran versatilidad.

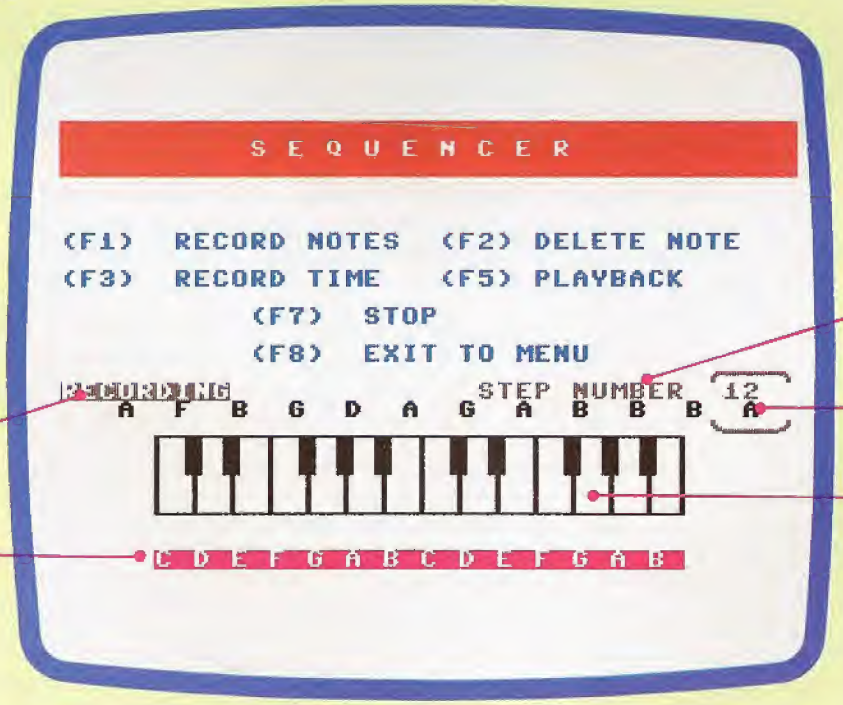
Otra dificultad que surge al tocar en "tiempo real" es que los parámetros tales como voz y octava no se pueden cambiar mientras el usuario está en "modalidad reproducción". Ello significa que, de hecho, el usuario queda limitado a dos octavas.

Con Music Maker se suministran dos manuales. La guía para el usuario es un folleto con instrucciones para cargar y una sucinta explicación sobre cómo se utilizan cada una de las diversas funciones. No es detallado, pero cumple con su cometido de iniciar al principiante. A partir de ese momento las instrucciones activadas por menú son suficientes para permitir que el usuario desarrolle las capacidades completas del paquete. El otro folleto, *Start playing keyboard*, contiene breves instrucciones acerca de cómo manejar teclados y una explicación de notación musical; incluye, además, las partituras de 28 populares canciones.

El Commodore Music Maker es, ciertamente, un valioso intento por hacer un uso cabal de las facilidades para sonido del Commodore 64. Para quien sea nuevo en el campo de la música por ordenador, el paquete es muy fácil de utilizar y, una vez que uno se acostumbra al teclado, es fácil tocar melodías. Los usuarios más avanzados quizá encuentren que el paquete carece de versatilidad y que el teclado está un tanto apiñado como para permitir efectos complicados. No obstante, una inversión que merece la pena para cualquiera que desee investigar las posibilidades de la música por ordenador.

Menú musical

Esta pantalla muestra la visualización de menú para la opción secuenciador del Music Maker



El menú muestra las seis opciones disponibles. Observe que las notas a tocar y su duración (TIME) se entran por separado

Recuerda al usuario que el Music Maker está en modalidad de grabación y no en la de reproducción

Esta línea le recuerda al usuario los nombres de las notas del teclado (C: *do*; D: *re*; E: *mi*; F: *fa*; G: *sol*; A: *la*; B: *si*)

Un cursor azul indica en el teclado qué nota se está tocando en cada momento

Éstas son las notas que se han entrado hasta ahora

El número de paso indica cuántas notas se han entrado hasta el momento en la memoria del secuenciador

Sistema público

Vamos a perfilar los tres sistemas de acceso público más importantes de Gran Bretaña: Telecom Gold, Prestel y Compunet

Los sistemas de acceso público se dividen en dos tipos: los de acceso público gratuitos, por cuya utilización no se le cobra al público ningún cargo, y los sistemas comerciales. Los tres sistemas comerciales más importantes de Gran Bretaña son Telecom Gold, Prestel (que incorpora el Micronet 800) y Compunet. Los sistemas de acceso público gratuito se componen normalmente como tablones de anuncios. Están operados aficionados, en micros normales, para beneficio de otros aficionados. Aquí nos centraremos en los sistemas comerciales y en el próximo capítulo examinaremos con detalle los sistemas de acceso público gratuitos.

Telecom Gold es el propio sistema de correo electrónico de la British Telecom. Dirigido fundamentalmente a los usuarios de gestión, ofrece correo electrónico instantáneo entre usuarios Gold, acceso a facilidades de télex y almacenamiento de archivos. Comprar un "buzón" cuesta 100 libras esterlinas y al cabo del primer mes (que es gratuito) al suscriptor se le cobra tanto por tiempo de permanencia en el sistema como por unidades de almacenamiento. La tarifa mínima es de 10 libras mensuales. La tasa por hora es elevada durante las horas de oficina, disminuyendo su precio alrededor de un tercio en cualquier otro momento. Las tarifas de almacenamiento se hacen por correo sin leer y tanto el correo como los archivos de trabajo son guardados en disco.

Dado que el Gold está diseñado pensando en las grandes empresas, las "contraseñas" de los usuarios se componen de un código de tres letras que identifica a la empresa (conocida como la "familia") y un código de tres dígitos que identifica a la persona dentro de la empresa. Así, por ejemplo, TCG035 identifica a Lucy Storer, de Telecom Gold, quien opera uno de los buzones de "ayuda".

Varias empresas de ordenadores han cogido una gran cantidad de buzones con el sistema de código de tres dígitos y tres letras, a tarifas reducidas, y han vendido luego buzones individuales a sus clientes también a tarifas reducidas. De este modo, los clientes de Tandy pueden adquirir un buzón TCC (Tandy Computer Corporation) por 20 libras en vez de 100. Al adquirirlo de esta manera uno no goza del uso gratuito durante el primer mes; pero, a menos que usted esté seguro de utilizar tarifas por más de 80 libras al mes, conviene averiguar si el fabricante de su micro ofrece una operación similar.

El Prestel también lo lleva British Telecom y, al igual que el Gold, ofrece facilidades de correo electrónico, aunque está diseñado básicamente para proporcionar información. Prestel es una gigantesca base de datos que abarca temas tan diversos como precios de acciones de bolsa, cocina, viajes, listas de éxitos de software, clubs de usuarios de ordenadores, actividades de compañías e informes meteorológicos locales.

El servicio opera según una estructura de árbol: uno empieza en el menú "raíz" y selecciona un área general; informática, por ejemplo. La selección se va estrechando gradualmente a través de una serie de submenús, hasta hallar lo que se estaba buscando: BBC Micro, software, software gratuito, juegos, Pacman. No obstante, si se conoce el número de "página" de la información buscada, pueden saltarse todos los menús e ir a ella directamente.

La principal área de interés para los usuarios aficionados a los ordenadores es la Micronet 800. Ésta permite participar en juegos en línea, cargar software (tanto gratuito como comercial), plantear preguntas técnicas y recibir respuestas, y leer noticias y análisis sobre micros. También puede enviar correo electrónico privado, si bien esta facilidad está relativamente poco perfeccionada.

El más rápido

El costo que representa enviar una carta de una página desde Londres a Glasgow mediante Red Star es de unas 7 libras esterlinas y tarda aproximadamente 6 ½ horas

La misma carta, enviada por correo First Class, tardaría 24 horas (aunque ha habido casos en los cuales han demorado varios días) y cuesta 17 peniques

El envío de la carta por télex es instantáneo, pero ruidoso. La maquinaria es costosa y el alquiler de la línea vale 88 libras por trimestre. Se han de abonar 14 peniques por cada 100 caracteres transmitidos

La misma carta enviada a través de Telecom Gold, sin embargo, se transmite instantáneamente. Costaría 10,5 peniques si se la enviara en horas de oficina o punta, y 3,5 a las horas de tarifa reducida. A ello habría que sumarle el costo de una llamada telefónica de un minuto

Kevin Jones

Hoja de datos de comunicaciones

Sistema	Telecom Gold	Prestel	Compunet	PSS
Número teléfono (datos)	01-278 4355, o a través del PSS	Varios (según la zona) 300/300 baud.=01–248 5747	Varios (según la zona)	Varios (según la zona)
Número teléfono (preguntas)	01-403 6777	01-278 3143	0536-205252	100 (pedir por Freefone PSS)
Costo	£100, más £10 por mes (mínimo)	Alrededor de £100 más £13 trimestrales (mínimo)	£100 más £30 por año	£25 más £25 por trimes (mínimo)
Horas funcionam.	24 horas	24 horas	24 horas	24 horas
Veloc. (baudios)	300/300 y 1200/1200	1200/75 (y 300/300 en Londres)	1200/75	300/300 y 1200/1200
Paridad	Ninguna	Videotexto	Videotexto	Ninguna
Bits de datos	8	Videotexto	Videotexto	8
Bits de final	1	Videotexto	Videotexto	1
Para obtener ayuda	INFO INFO	*0£ (*36£ para dejar mensaje)	Seleccionar HELP del menú	Ninguna disponible
Para desconectarse	OFF	Colgar	Seleccionar LEAVE del menú	Colgar
¿Correo electrón.?	Sí	Sí	Sí	No
¿Mensajes públicos?	Sí	Sí	Sí	No
¿Software comercial?	No	Sí	Sí	No
¿Software gratuito?	No	Sí	Sí	No

Para unirse al Micronet se necesita un modem adecuado y software para comunicaciones, y los usuarios del BBC Micro, el Spectrum o el Commodore 64 pueden obtenerlos de Micronet 800.

Compunet ofrece una gama de servicios similar a la que proporciona el área Micronet 800 del Prestel. El usuario puede enviar correo electrónico, dejar mensajes al público, cargar software y acceder a noticias e información relativa a micros.

Para acceder a Compunet se necesita un modem Commodore. Éste contiene una ROM que contiene el software de comunicaciones necesario para conectar con el sistema, y un número de serie exclusivo. Cuando el suscriptor establece conexión se le solicita su identificación, que se compara con el número de serie de la ROM de su modem: si no coinciden, Compunet no le permitirá el acceso.

Este sistema posee a la vez ventajas e inconvenientes. La ventaja obvia tanto para Compunet como para el usuario es que no importa que alguien le "robe" su código de identificación, porque no podrá utilizarlo sin su modem. La desventaja es que el usuario no puede conectar con el sistema a través del modem de alguna otra persona.

El sistema incorpora otro dispositivo de protección pensado para impedir la piratería de software. Cuando el suscriptor carga software desde Compunet, en el mismo se "estampa" el número de serie del modem utilizado para cargarlo. El software sólo se podrá ejecutar con el modem enchufado.

Al igual que el Prestel, el Compunet se basa en una estructura arborescente, con menús principales que llevan a submenús. El Compunet es, sin embargo, más "amable" que el Prestel, porque ofrece menús en el área principal de la pantalla y un menú de instrucciones que se va desplazando en la parte inferior. El menú de instrucciones ofrece opciones tales como *"VIEW highlighted option"* (ver opción resaltada), *"Go back to the LAST page"* (retroceder hasta la última página), *"COPY the current page to tape or disk"* (copiar la página en curso a cinta o disco), etc. Se utilizan las teclas del cursor arriba/abajo para realizar la selección del menú Compunet, y las teclas izquierda/derecha para hacer la selección del menú de instrucciones.

El Commodore Communications Modem cuesta 100 libras (incluyendo el software) y la tarifa de suscripción a Compunet es de 30 libras al año, siendo gratuitos los primeros 12 meses. El acceso cuesta 7 libras por hora en horario de oficina, siendo gratuito el resto del tiempo (tardes, noches y fines de semana).

El Compunet no es (contrariamente a lo que suele pensar la gente y según la impresión que da la propia Commodore) el servicio de videotexto propio de Commodore. Lo lleva Compunet Teleservices Ltd., una empresa independiente. En la actualidad el sistema sólo soporta al Commodore 64, pero la intención de Compunet es ir soportando de forma gradual todos los micros populares.

Packet Switching Service

Por último, examinemos brevemente el Packet Switching Service (PSS), de la British Telecom, que no es un servicio por ordenador en el sentido del Gold, el Prestel o Compunet, sino una forma económica de acceder a otros sistemas de ordenador.

A menos que usted se encuentre en la zona de llamadas locales del servicio de ordenador que desea utilizar, el principal gasto puede ser el costo de la llamada telefónica. Esto es particularmente cierto si utiliza sistemas norteamericanos, pero también se aplica a las llamadas nacionales de larga distancia. El PSS es la respuesta de la British Telecom a este problema.

El PSS es una red conectada a los principales sistemas de ordenador de acceso público de todo el mundo. Los usuarios pueden conectarse a un "nodo" (punto de entrada) PSS local y acceder, a través del mismo, a cualquiera de los sistemas conectados. El usuario sólo paga el costo de la llamada al nodo local (normalmente, una llamada local) más una tasa PSS. El recuadro "Sesión tipo utilizando el Telecom Gold" muestra cómo se accede al Gold a través del PSS.

Cuando obtenga el tono de portadora, pulse <CR> <CR> a2 <CR>. Esto indica al PSS que Ud. es un terminal CRT, ASCII estándar. PSS responderá entonces con un trozo de código, que será algo parecido a esto:

SLO\A01-7531740139

ntlgold <+non-printing password>

No se le invitará a conectarse; ¡ni siquiera se le ofrecerá un carácter de aviso! El PSS simplemente aguardará a que entre su identificación de usuario de la red (NUI). Ésta es "n" seguido de un ID y una contraseña, ambos de 6 caracteres. Ud. entra todo en una sola línea, que no se visualizará:

ADD?

Si el PSS no reconoce su ID y su contraseña, le responderá con un "NUI?" y esperará que vuelva a probar. Cuando esté conectado, le pedirá la dirección de red del servicio que desea mediante el mensaje:

Luego digite la dirección del servicio que Ud. desea. Ésta será una "a" seguida por un código de 11 dígitos. En este caso, queremos Tel. Gold; su código es

a21920100481

El PSS intenta conectarnos con este servicio. Si todo va bien, visualizará algo así como:

23421920100481+COM

Primecom Network 18.4G System 81

Y Ud. ya está en el sistema pedido. Hemos llamado al Telecom Gold, de modo que veamos cómo funciona:

Please Sign On

>id tcc007

Para conectarse (SIGN ON) hemos de entrar las letras "id" seguidas por una identidad de seis caracteres. Este ID tendrá tres letras describiendo la "familia" de buzones (ver texto principal) y tres números identificando el buzón individual:

Password: <non-printing password>

Gold le pedirá entonces la contraseña. Esta está compuesta por entre 1 y 8 caracteres alfanuméricos, según el caso. El PSS visualiza

Gold lo conecta en:

TELECOM GOLD Automated Office Services 18.4G(81)
On At 18:23 30/09/84 BST
Last On At 16:53 27/09/84 BST

Mail call (1 Unread)

El Gold le dice si en su buzón hay alguna carta y si Ud. ya la ha leído:

mail
Send, Read or Scan: read

Cada carta le dará su número, la fecha y hora en que se envió. Después le pregunta si desea leerla, presentándole un mensaje "MORE?" Pulse <CR> para continuar, "n" para pasar por alto

Para leer su correspondencia, entre:

To: TCC007 (81, TCC007)
From: L STORER (BTG035) Posted: Fri 28-Sep-84 16:46 BST UK Sys 80
Subject: Reply to: query re 81 to 84 changeover

—MORE— <CR>

YES, WHEN I HAVE ALL THE DETALLS I WILL CERTAINLY REPLY

LUCY

Se le preguntará qué desea hacer con la carta. Ud. puede guardarla (SAVE) en sus archivos personales, contestarla (REPLY), eliminarla (DELETE) o dejarla donde está. Se le cargará una tarifa de almacenamiento por las cartas sin leer. Si desea un almacenamiento gratuito, ¡escríbase una carta a Ud. mismo y después léala!

Action Required: reply
Text:

I see. The Helpline tells me that Gold will be doing a mail out to all users informing them of the change. Can you tell me when this will be happening? Will everyone on all systems be notified? If so, it shouldn't be necessary for me to have to tell them personally.

Surya

Ahora enviemos la carta:

send
BTG035 -- Sent

Ahora el Gold nos pregunta qué más deseamos hacer con la carta. Dado que ya hemos acabado, la vamos a borrar:

Action Required: delete
End of Mail

Pulsando <CR> se ahorra la mayoría de los mensajes no deseados:

Send, Read or Scan: < CR >

>mailck
No mail at this time

Si Ud. ha permanecido en el sistema durante un rato, es una buena idea ver si mientras estaba conectado ha llegado alguna otra carta:

Hemos acabado; desconectémonos:

> off
Off At 18:29 30/09/84 BST
Connect Mins = 7
Compute Secs = 4/3

CLR DTE (00) 00:00:06:07 47 49

Ahora estamos otra vez en el PSS, pero ante este mensaje Ud. puede simplemente colgar.

Ian McKinnell

Hundirse o nadar

Nos corresponde analizar el diseño de los dos últimos escenarios especiales de "El bosque encantado": el pantano y el poblado

Aún hemos de considerar dos de los escenarios especiales de *El bosque encantado*: el pantano y el poblado. Comencemos por el pantano. Al igual que para todos los escenarios especiales, es importante decidir sobre una línea argumental antes de ponernos a programar. Esta historia puede ser tan compleja o tan simple como le parezca conveniente al programador, pero ha de tener presente que una trama muy densa para cada escenario puede suponer un enorme esfuerzo de programación y devorar grandes agujeros de la memoria residual del ordenador.

El pantano

La trama argumental para el pantano es la siguiente:

Cuando el jugador entre en el pantano empieza a hundirse. Puede utilizar todas las "instrucciones normales" disponibles, pero no puede salir de la ciénaga.
En cambio, el jugador debe optar por nadar en caso de que quiera salir del pantano.
El jugador sólo puede nadar si lleva consigo menos de dos objetos, puesto que éstos aumentan su peso, contribuyendo a hundirlo.
Si el jugador transporta consigo dos objetos, ha de abandonar uno si no quiere hundirse.
Todos los objetos que se abandonen en el pantano se pierden para siempre.

```
4870 REM **** S/R PANTANO ****
4875 SF=1
4880 SN$="COMIENZAS A HUNDIRTE EN EL PANTANO.":GOSUB 5500
4885 PRINT:INPUT"INSTRUCCIONES";IS$
4890 GOSUB2500:REM DESCOMPONER INSTRUCCION
4895 IF F=0 THEN 4885:REM NO VALIDA
4900 GOSUB3000:REM INSTRUCCIONES NORMALES
4910 IF VB$="MIRAR" THENGOSUB2000:GOTO4885
4915 IF VB$="DEJAR" THEN IV$(F,2)="-2":REM OBJETO PERDIDO
     PARA SIEMPRE
4917 IF VF=1 THEN 4885:REM INSTRUCCION NORMAL
4920 REM ** INSTRUCCIONES NUEVAS **
4925 IF VB$<>"NADAR"THEN SN$="NO COMPRENDO":GOSUB5500:
     GOTO4885
4930 REM ** NADAR **
4932 F=0
4935 FOR I=1 TO 2
4940 IF IC$(I)<>"" THEN F=F+1
4950 NEXT I
4955 IF F<2 THENGOSUB5035:RETURN:REM ALEJARSE NADANDO
4960 GOSUB 5000:RETURN:REM LLEVANDO DOS OBJETOS
5000 REM **** S/R LLEVANDO DOS OBJETOS ****
5010 SN$="LOS OBJETOS SE EMPUJAN HACIA ABAJO Y TE VAS
     HUNDIENDO.":GOSUB5500
5012 PRINT:INPUT"INSTRUCCIONES";IS$
5015 GOSUB2500:REM DESCOMPONER INSTRUCCION
5020 IF VB$<>"DEJAR" THENGOSUB5080:REM HUNDIR
5025 GOSUB 3900:IV$(F,2)="-2":REM ABANDONAR OBJETO
5030 IF HF=0 OR F=0 THEN 5080:SEM HUNDIR
5035 REM **** ALEJARSE NADANDO ****
5040 SN$="AHORA PUEDES CRUZAR EL PANTANO A NADO. EN QUE
     DIRECCION IRAS?":GOSUB5500
5050 SL$(2)="00080605":GOSUB2300:REM DEFINIR Y VISUALIZAR
     SALIDAS
5055 PRINT:INPUT"INSTRUCCIONES";IS$
5060 GOSUB2500:REM DESCOMPONER INSTRUCCION
5062 IF F=0 THEN 5055:REM NO VALIDA
5065 GOSUB3500:REM MOVER
5067 SL$(2)="00000000";REM DATOS SALIDA CERO
5070 RETURN
5075 :
5080 REM **** S/R HUNDIR ****
5085 SN$="TE HUNDES EN EL PANTANO Y TE AHOGAS":GOSUB5500
5090 END
```

Las líneas 4870-4917 permiten manejar las instrucciones normales, empleando las subrutinas estándares diseñadas con anterioridad. Si el jugador elige dejar un objeto inmediatamente al entrar en el pantano, se ocupará de ello la rutina DEJAR. Sin embargo, esta rutina reintegra la posición del objeto dejado al inventario principal, IV$(,), utilizando el valor en curso del contador de escenarios, P. Ello a su vez significa que, por cuanto atañe al programa, el objeto dejado se halla ahora en el escenario del pantano. Si deseamos perder todo rastro de un objeto que sea abandonado en la ciénaga, en el caso de que un jugador desee dejar aquí un objeto, hemos de corregir la entrada correspondiente en IV$(,).

Recuerde que IV$(F,2) contiene la posición del objeto F. Normalmente éste es el número de escenario, o −1 si el objeto lo lleva consigo el jugador. Para lograr que el objeto desaparezca por completo del juego, hemos de hacer que IV$(F,2) no pueda ser interpretada como un escenario ni indicar que el jugador tiene el objeto. En la línea 4915, IV$(F,2) toma el valor −2, habiéndosele ya asignado a F el objeto en cuestión mediante la rutina DEJAR.

Si el jugador opta por NADAR, el programa baja hasta la línea 4930. Aquí se comprueba el inventario personal del jugador para determinar cuántos objetos lleva consigo. Si el contador es menor que dos, entonces el programa llama a la subrutina ALEJARSE NADANDO, que permite que el jugador salga del pantano. Si el jugador está transportando dos objetos, se le ofrece la oportunidad de deshacerse de uno; si no sigue este curso de acción, el jugador se hunde.

La rutina ALEJARSE NADANDO permite al jugador especificar en qué dirección quiere nadar. Los datos de salida que se dan para el pantano en las sentencias DATA originales son 00000000, que indican que la ciénaga no tiene salida. La línea 5050 vuelve a definir los datos de salida y llama a la subrutina que las describe. Al jugador se le permite entonces seleccionar una y, por consiguiente, huir del pantano. Observe que al final de esta rutina los datos de salida para el pantano se ponen a cero, de modo que si posteriormente el jugador volviera a entrar en la ciénaga, no habría salida.

El poblado

El poblado ofrece el camino para escapar del bosque encantado, si bien el jugador todavía debe lle-

var a cabo algunas tareas antes de que se le permita abandonar la aventura.

La línea argumental para el poblado es ésta:

El poblado está rodeado por una muralla alta y aparentemente imposible de escalar.
La puerta de la muralla está vigilada.
Para entrar en el poblado el jugador debe primero matar al guardián, utilizando la escopeta.
El jugador debe entonces abrir la cerradura de la puerta mediante la llave, para huir del bosque.

La rutina del poblado consta de dos partes: primero, se ha de sortear el peligro del guardián y, segundo, se ha de abrir la puerta. No es difícil imaginar un escenario en el cual el jugador, llevando la escopeta, llegue al poblado y mate al guardián, sólo para descubrir que se necesita una llave para abrir la puerta. Si el jugador no posee la llave, entonces habrá de abandonar el escenario del poblado en busca de ella. Si, al regresar el jugador, se ofrece la misma descripción del escenario del poblado (es decir, que hay un guardián custodiando la puerta), ello no será coherente. Se debe afrontar y resolver un problema particular del juego una sola vez. Si el guardián es muerto, entonces se debe suprimir esa "característica" del poblado.

Esto no es tan difícil como parece. Se puede emplear una bandera que señale si el guardián está vivo o muerto. Con este fin se utiliza la bandera GF: su valor inicial es cero, indicando que el guardián está vivo. Si el jugador mata al guardián, entonces GF se pone a 1. El valor de GF se puede comprobar al entrar en el escenario del poblado para determinar si aún existe el peligro del guardián o no. Habiendo muerto el guardián, el jugador avanza hacia la puerta.

Las instrucciones se pueden descomponer y manejar mediante las subrutinas estándares que ya poseemos. Si el jugador posee la llave y la utiliza para abrir la puerta, habrá completado con éxito la aventura.

```
5100 REM **** S/R POBLADO ****
5102 SF=1
5105 SN$="EL POBLADO ESTA RODEADO POR UNA ALTA MURALLA
     .":GOSUB5500
5106 IF GF<>0 THEN GOSUB5190:RETURN:REM PUERTA
5107 SN$="HAY UN GUARDIAN JUNTO A LA PUERTA DE ACCESO AL
     POBLADO":GOSUB5500
5115 PRINT:INPUT"INSTRUCCIONES";IS$
5120 GOSUB2500:IF F=0 THEN 5115:REM NO VALIDA
5125 GOSUB3000:REM INSTRUCCIONES NORMALES
5130 IF VB$="MIRAR" THEN GOSUB2000:REM DESCRIBIR
5135 IF VB$="AVANZAR" AND MF=1 THEN RETURN
5140 IF VF=1 THEN 5115:REM SIGUIENTE INSTRUCCION
5145 IF VB$<>"MATAR" THEN SN$="NO COMPRENDO":GOSUB-
     5500:GOTO5115
5150 REM ** MATAR **
5155 SN$="QUE USARAS PARA MATAR AL GUARDIAN?":
     GOSUB5500
5160 SN$="ENTRE OBJETO 0 <I> PARA INSTRUCCION":
     GOSUB5500
5162 INPUT IS$:IF IS$="I" THEN 5115
5165 GOSUB2500:REM DESCOMPONER
5167 IF F=0 THEN 5160:REM NO VALIDA
5170 GOSUB5300:IF F=0 THEN SN$="NO HAY NINGUNA
     "+WS:GOSUB5500:GOTO5160
5172 OV=F:GOSUB5450:REM LLEVA CONSIGO EL OBJETO
5174 IF HF=0 THEN SN$="TU NO TIENES
     LA "+IV$(F,1):GOSUB5500:GOTO5160
5175 IF F<>1 THEN SN$="LA "+IV$(F,1)+" NO SIRVE":GOSUB-
     5500:GOTO5160
5180 SN$="MATAS AL GUARDIAN":GOSUB5500:GF=1
5185
5190 REM **** S/R PUERTA CERRADA CON LLAVE ****
5195 SN$="DEBES AVANZAR Y TRATAR DE ABRIR LA PUERTA DE
     ENTRADA AL POBLADO"
5200 SN$=SN$+" PERO LA PUERTA ESTA CERRADA Y NO SE MUEVE
     ":GOSUB5500
5205 PRINT:INPUT"INSTRUCCIONES";IS$
5210 GOSUB2500:IF F=0 THEN 5205:REM NO VALIDA
5215 GOSUB 3000:REM INSTRUCCIONES NORMALES
5220 IF VB$="MIRAR" THEN GOSUB2000:REM DESCRIBIR
5225 IV VB$="AVANZAR" AND MF=1 THEN RETURN
5230 IF VF=1 THEN 5205:REM SIGUIENTE INSTRUCCION
5232 IF VB$="USAR" THEN 5240
5234 IF VB$="ABRIR"THEN SN$="COMO?":GOSUB5500:
     GOTO5205
5235 SN$="NO COMPRENDO":GOSUB5500:GOTO5205
5240 GOSUB5300:REM OBJETO VALIDO
5242 OV=F:GOSUB5450:REM LLEVA OBJETO CONSIGO
5244 IF F=0 THEN SN$="NO HAY
     NINGUNA "+W$:GOSUB5500:GOTO5205
5246 IF HF=0 THEN SN$=TU TIENES LA
     "IV$(F,1):GOSUB5500:GOTO5205
5248 IF F<>3 THEN SN$="LA "+IV$(F,1)+" NO SIRVE":GOSUB-
     5500:GOTO5205
5250 REM ** ATRAVESAR LA PUERTA Y A SALVO **
5255 SN$="ABRES LA PUERTA Y, DISFRAZANDOTE CON
     LA ROPA"
5260 SN$=SN$+" DEL GUARDIAN MUERTO, ATRAVIESAS EL
     POBLADO SIN SER DESCUBIERTO"
5265 SN$=SN$+" Y LLEGAS A LA SEGURIDAD DEL MUNDO
     EXTERIOR.":GOSUB5500
5270 END
```

Para poder llamar a estas dos rutinas de escenario especial debemos modificar la línea 2720 de la subrutina que decide si un escenario es o no especial. Introduzca esta modificación:

```
2720 ON P GOSUB4590,4870,5100,4590
```

Complementos al BASIC

Spectrum

En el listado de *El bosque encantado*, sustituya SN$ por S$, IS$ por T$, VB$ por B$, IV$(,) por V$(,), SL$() por X$() e IC$() por I$().
Reemplace las siguientes líneas:

```
2720 IF P=1 THEN GOSUB 4590
2722 IF P=2 THEN GOSUB 4870
2724 IF P=3 THEN GOSUB 5100
2726 IF P=4 THEN GOSUB 4590

4937 LET A$=IC$(I):GOSUB7000
4940 IF A$<>"" THEN LET F=F+1

5175 IF F<>1 THEN LET S$="LA":
     LET A$=V$(F,1):GOSUB7000
5177 IF F<>1 THEN LET S$=S$+"NO
     SIRVE":GOSUB5500:GOTO5160

5248 IF F<>3 THEN LET S$="LA"
     :LET A$=V$(F,1):GOSUB7000
5249 IF F<>3 THEN LET S$=S$+"NO
     SIRVE":GOSUB5500:GOTO5205
```

En el listado de *Digitaya*, reemplace SN$ por S$, IS$ por I$, VB$ por B$, IV$(,) por V$(,) e IC$() por I$(). Sustituya estas líneas:

```
3650 LET S$="TU ":LET A$=V$(F,1):
     GOSUB 7000
3655 LET S$=S$+" NO SIRVE DE
     NADA, LA FUERZA AUMENTA"

4520 LET V$(4,2)=STR$(INT(RND(1)
     *40+8)
```

BBC Micro:

En el listado de *El bosque...* sustituya esta línea:

```
190 LET GF=0
```

En el listado de *Digitaya* reemplace esta línea:

```
4520 IV$(4,2)=RND(40)+8
```






Listado de "Digitaya"

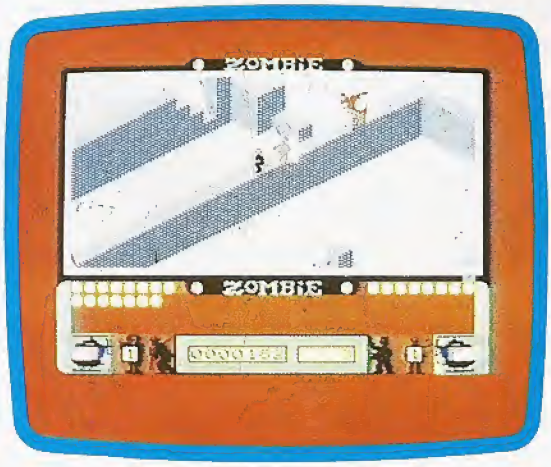
```
2960 REM **** S/R PUERTA PARA EL USUARIO ****
2970 SF=1
2980 SN$="EL ESCAPE ESTA A TU ALCANCE PERO EL VENDEDOR
     DE BILLETES DEL RDD"
2990 SN$=SN$+" TE IMPIDE EL PASO, TE DICE QUE TIENE
     INSTRUCCIONES DE"
3000 SN$=SN$+" ACEPTAR SOLO ENTRADAS. NO OBSTANTE
     ACEPTA LAS PRINCIPALES"
3010 SN$=SN$+" TARJETAS DE CREDITO."
3020 GOSUB 5880:REM FORMATEAR SALIDA
3030 :
3040 PRINT:INPUT"INSTRUCCIONES";IS$
3050 GOSUB1700:REM ANALIZAR INSTRUCCIONES
3060 GOSUB 1900:REM ACCIONES NORMALES
3070 IF MF=1 THEN RETURN:REM IRSE
3080 IF VF=1 THEN 3040:REM SIGUIENTE INSTRUCCION
3090 IF VB$<>"DAR" THENPRINT"NO COMPRENDO":GOTO3040
3100 :
3110 REM ** LA INSTRUCCION ES DAR **
3120 GOSUB5730:REM ES VALIDO EL OBJETO
3130 IF F=0 THEN PRINT"NO HAY NINGUN ";NN$:GOTO3040:
     REM SIGUIENTE INSTRUCCION
3140 :
3150 REM ** ES EL OBJETO TARJETA DE CREDITO **
3160 IF F<>5 THEN PRINT"SOLO ACEPTA TARJETAS DE
     CREDITO":GOTO3040
3170 :
3180 REM ** LLEVA TARJETA CONSIGO **
3190 OV=5:GOSUB5830
3200 IFHF=0THENPRINT"NO TIENES LA ";IV$(5,1):GOTO 3040
3210 :
3220 SN$="EL EMPLEADO COGE LA TARJETA Y DICE 'ESTA
     SERVIRA, SEÑOR'"
3230 GOSUB5880:REM FORMATEAR SALIDA
3240 SN$="SE TE PERMITE PASAR LA BARRERA Y ENTRAR EN
     LA PUERTA PARA EL USUARIO"
3250 GOSUB5880:REM FORMATEAR SALIDA
3260 :
3270 REM ** LLEVA CONSIGO EL DIGITAYA **
3280 OV=6:GOSUB5830
3290 IF HF=1 THEN 3380:REM EXITO
3300 :
3310 REM ** FRACASO **
3320 SN$="BIEN HECHO HAS LOGRADO ESCAPAR DE LAS
     GARRAS"
3330 SN$=SN$+" DE LA MAQUINA, PERO HAS FRACASADO EN
     TU MISION"
3340 SN$=SN$+" CONSISTENTE EN RECUPERAR EL
     MISTERIOSO DIGITAYA"
3350 GOSUB5880:REM FORMATEAR SALIDA
3360 END
3370 :
3380 REM ** EXITO **
3390 SN$="FELICITACIONES, HAS CUMPLIDO CON TU MISION"
3400 SN$=SN$+" DE RESCATAR AL MARAVILLOSO DIGITAYA
3410 SN$=SN$+" DE LAS GARRAS DE LA MAQUINA."
3420 GOSUB5880:REM FORMATEAR SALIDA
3430 END
3440 :
3450 REM **** S/R PUERTA PARA CASSETTE ****
3460 SF=1
3470 SN$="SIENTES UNA FUERZA IRRESTIBLE QUE TE EMPUJA
     HACIA"
3480 SN$=SN$+" UNA SUSPENSION MAGNETICA
     PERMANENTE"
3490 GOSUB5880:REM FORMATO
3500 NS=0:REM COMENZAR A CONTAR LAS INSTRUCCIONES
3510 REM ** INSTRUCCIONES **
3520 NS=NS+1:IFNS>3THEN3770:REM ABSORBIDO
3530 PRINT:INPUT"INSTRUCCIONES";IS$
3540 GOSUB1700:REM ANALIZAR INSTRUCCIONES
3550 GOSUB1900:REM ACCIONES NORMALES
3560 IF MF=1 THEN MF=0:PRINT"NO PUEDES
     MOVERTE...TODAVIA":GOTO3510
3570 IF VF=1 THEN 3510:REM SIGUIENTE INSTRUCCION
3580 IF VB$<>"USAR" THENPRINT"NO COMPRENDO":
     GOTO3510
3590 REM ** LA INSTRUCCION ES USAR **
3600 GOSUB5730:REM ES VALIDO EL OBJETO
3610 IF F=0 THENPRINT"NO HAY NINGUN ";NN$:GOTO3510
3620 :
3630 REM ** ES EL OBJETO EL ACTIVADOR DEL BUFFER **
3640 IF F=8 THEN3680:REM OK
3650 SN$="TU "+IV$(F,1)+" NO SIRVE DE NADA, LA FUERZA
     SE INTENSIFICA"
3660 GOSUB5880:GOTO3510:REM SIGUIENTE INSTRUCCION
3670 :
3680 OV=8:GOSUB5830:REM LLEVA CONSIGO ACTIVADOR
     BUFFER
3690 IF HF=0THEN SN$="NO TIENES EL "+IV$(8,1):
     GOSUB5880:GOTO3510
3700 :
3710 REM ** SALVADO **
3720 SN$="UTILIZAS EL ACTIVADOR DEL BUFFER PARA
     CONTRARRESTAR EL EMPUJE"
3730 SN$=SN$+" HACIA EL OLVIDO MAGNETICO. LA FUERZA
     DISMINUYE"
3740 GOSUB5880:REM FORMATEAR
3750 RETURN
3760 :
3770 REM ** ABSORBIDO **
3780 SN$="LA FUERZA SE VUELVE DEMASIADO INTENSA Y ERES
     EMPUJADO"
3790 SN$=SN$+" A TRAVES DE LA PUERTA PARA CASSETTE A
     LA NADA MAGNETICA."
3800 GOSUB5880:REM FORMATEAR
3810 END
4180 REM **** S/R DISPOSITIVO TRIESTADO ****
4190 SF=1
4200 SN$="UN ENORME CARTEL REZA 'E/S POR AQUI' PERO
     CUANDO TE ESTAS ACERCANDO AL MISMO"
4210 SN$=SN$+" UN REVISOR GRITA 'BILLETES POR FAVOR'"
4220 GOSUB5880:REM FORMATEAR
4230 :
4240 REM ** INSTRUCCIONES **
4250 PRINT:INPUT"INSTRUCCIONES";IS$
4260 GOSUB1700:GOSUB1900:REM ANALIZAR
4270 IFMF=1 THEN RETURN
4280 IFVF=1 THEN4240:REM SIGUIENTE INSTRUCCION
4290 IF VB$<>"DAR"ANDVB$<>"OFRECER"THEN PRINT"NO
     COMPRENDO":GOTO4240
4300 REM ** LA INSTRUCCION ES DAR **
4310 GOSUB5730:REM ES VALIDO EL OBJETO
4320 IFF=0THENPRINT"NO HAY NINGUN ";NN$:GOTO4240:REM
     SIGUIENTE INSTRUCCION
4330 :
4340 REM ** ES UN TICKET EL OBJETO **
4350 IFF=4 THEN4400:REM OK
4360 SN$="EL REVISOR SACUDE LA CABEZA Y DICE"
4370 SN$=SN$+" 'NO PUEDO ACEPTAR ESTE' +IV$(F,1)
4380 GOSUB5880:GOTO4240:REM SIGUIENTE INSTRUCCION
4390 :
4400 OV=4:GOSUB5830:REM LLEVA EL BILLETE CONSIGO
4410 IFHF=0THEN PRINT"NO TIENES EL BILLETE":GOTO4240
4420 :
4430 REM ** OK **
4440 SN$="EL REVISOR ACEPTA TU BILLETE Y TE PERMITE"
4450 SN$=SN$+" TRASPASAR LA BARRERA."
4460 GOSUB5880:REM FORMATEAR
4470 REM ** ELIM. BILLETE DE LA LISTA **
4480 F=0
4490 FORJ=1T04
4500 IF IC$(J)=IV$(4,1)THENIC$(J)="":J=4
4510 NEXT J
4520 IV$(4,2)=STR$(INT(RND(TI)*40+8)):REM REASIGNAR
     POSICION BILLETE
4530 P=15:MF=1:RETURN
```

Yo, el jurado

Nuestros comentaristas hacen una valoración de dos programas de juegos escritos para el Sinclair Spectrum

Zombie zombie

Quicksilva



Dave Nicholls: ¡Oh, no! Antescher ha sido invadida por los zombies, zombies verdes y horribles que se ponen rojos por la furia y te atacan cuando te acercas demasiado a ellos.

Para ayudarte a librar a la ciudad de esta marea de fanáticos de Duran Duran, dispones de un helicóptero con el cual puedes volar con total seguridad. Pero debes abandonarlo para destruir a los zombies, ¡hazaña que se consigue cogiéndolos por detrás! Cuando les haces blanco, se convierten en una especie de esclavos y los puedes conducir hasta cualquier pared, desde donde saltarán hacia la muerte al son de *Ten green bottles* (Diez botellas verdes). Si no estás muy seguro de poder controlar a los zombies de este modo, puedes dispararles con tu *puffer*, lo que les alejará hasta una distancia segura.

Básicamente se trata de otro *Ant attack* (Ataque de las hormigas) y por ello les gustará a los usuarios que disfrutaron con este juego. Pero si *Ant attack* te resultó una experiencia soporífera, no te molestes en probar el *Zombie zombie*. No obstante, si no has probado ninguno de los dos, probablemente el mejor de ambos sea el que ahora comentamos. *3/5*
ÉXITO

Ross Holman: Es demasiado parecido a *Ant attack* como para ofrecer nada emocionante. Las melodías son atrayentes y te lo pasarás bien escribiendo *graffiti* en los ladrillos. *1/5*
FIASCO

Roger Willis: Perseguir a los resucitados puede ser divertido, aun cuando sea un entretenimiento de escasa originalidad. *3/5*
ÉXITO

Eddie Kidd jump challenge

Martech

Roger Willis: Puedo afirmar, por mi propia experiencia, que inclinarte hacia adelante o hacia atrás cuando estás saltando en motocicleta puede afectar seriamente tu estabilidad. También puedo asegurar que, tal como sucede en este programa, hacerlo mal supone unas consecuencias desastrosas.

La distancia de aproximación parece irrelevante, dado que al despegar de la rampa la motocicleta ha puesto sus cinco marchas y la velocidad indicada es la máxima. En realidad, todo lo que tienes que hacer es controlar la moto en el aire y eso, presumiblemente, determina la cantidad de coches sobre los que consigues saltar con éxito. Suena bien, ¿no? Bueno, pues no es así...

Los mezquinos gráficos sólo sirven para subrayar el hecho de que Eddie Kidd le ha dado su nombre a un juego recreativo bastante primitivo que se vuelve aburrido antes de que el jugador consiga dominarlo. La pantalla de introducción (o de práctica) exige que saltes sobre barriles de gasolina; ¡lo malo es que tus saltos siempre son buenos! Oh, sí, más allá de un cierto límite los saltos producen una invitación a SAVE (guardarlos) en una cinta de cassette virgen con el objeto de participar en competiciones; pero, yo, en realidad, no creo que nadie se moleste en hacerlo. *1/5*
FIASCO

Ross Holman: Puedes controlar a Eddie bastante bien; pero incluso después de cierta práctica me resultó difícil realizar saltos con éxito en distancias relevantes. ¡Las "ráfagas de viento" tampoco me resultaron de ayuda! *2/5*
FIASCO

Para los más pequeños

Las primeras palabras de un niño
My Talking Computer está pensado como la primera introducción del niño al ordenador. Está diseñado para que su utilización sea lo más sencilla posible. Las cubiertas, incluidas en un libro de anillas, se colocan sobre un teclado sensible al tacto y se sujetan mediante un marco. En la ranura situada en el lado izquierdo se pueden insertar fácilmente módulos de ampliación que contienen programas extras

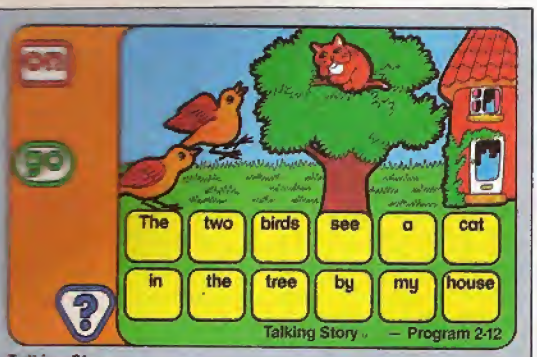
My Talking Computer es un "ordenador" destinado a la enseñanza de niños en edad preescolar

My Talking Computer, de Microspeech, está dirigido a niños a partir de tres años de edad, y la empresa afirma que su máquina ya se está utilizando en escuelas primarias de Gran Bretaña. El "ordenador" está alojado en una carcasa de plástico ligera que mide 23×25 cm. En un panel sensible al tacto situado en la parte delantera de la máquina se colocan cubiertas para los diversos programas, que se mantienen en su sitio mediante un marco. En este sentido se parece al teclado Touchmaster, aunque My Talking Computer tiene muchos menos puntos de contacto. Lleva instalado My Talking Clock que, utilizado junto con uno de los programas incorporados, enseña al niño a decir la hora.

Al costado de la máquina hay una ranura en la cual se insertan los módulos de los programas (dispositivos similares a cartuchos). Aunque My Talking Computer representa un enfoque nuevo e ingenioso para enseñar a niños pequeños, los programas residentes están obviamente limitados en cuanto a sus aplicaciones, lo cual acorta de manera inevitable la vida útil de la máquina en cuanto concierne al pequeño. No obstante, los fabricantes pretenden superar este problema proporcionando una ranura de ampliación que permite la instalación de ROM extras. A pesar de que la primera de estas ROM, Expansion Module One (módulo de ampliación uno), parece ser de un nivel apenas ligeramente superior al de la ROM existente, la intención del fabricante parece ser producir una serie que vaya sofisticándose a medida que el niño crezca. Encima de la ranura de ampliación hay un pequeño altavoz. La máquina opera mediante cinco pilas de 1,5 V o bien mediante alimentación eléctrica externa.

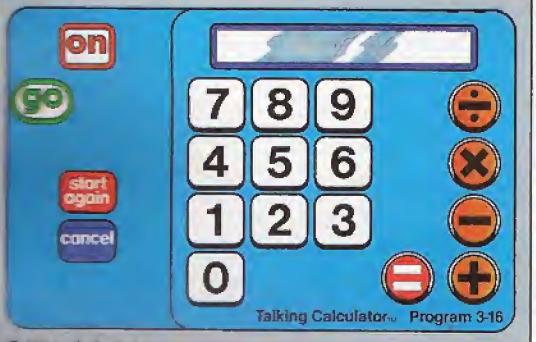
Los programas están almacenados en un único chip de ROM que, según Microspeech, contiene más de 120 K de datos, si bien el ordenador no cargará esta información de una sola vez. El software basado en ROM está dividido en cinco programas maestros, cada uno de los cuales está subdividido en varios otros basados en el mismo tema. El primer programa maestro está dedicado a enseñar aritmética, con ejercicios tales como reconocimiento de números y operaciones sencillas de suma, resta, multiplicación y división. El segundo enseña la relación entre imágenes y palabras. El tercero es una "calculadora parlante" que lee las cifras a medida que el niño las va entrando y luego dice el resultado. El cuarto programa consiste en juegos "parlantes" que comprueban la habilidad del pequeño para responder a las relaciones, expresadas en términos tales como "hallar el perro". El quinto programa, que ya hemos mencionado, es para aprender a decir la hora.

El niño utiliza un programa determinado remitiéndose a un libro de cubiertas con anillas e instalando la cubierta seleccionada debajo del marco situado en la parte frontal del ordenador. Pulsando el cuadrado ON de la cubierta la máquina se enciende y una voz femenina responde diciendo "Hello".

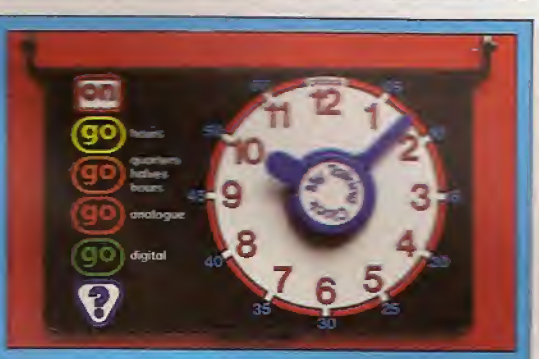


Talking Story
Esta cubierta está diseñada para niños de 2 a 12 años. El ordenador reconoce qué programa se está utilizando en función de las posiciones de los "botones" ON y GO. Cuando se pulsa un cuadrado con palabra, el ordenador la pronuncia

Sentence Maker
Este programa enseña a reconocer palabras. Cuando el niño pulsa el signo de interrogación, el ordenador dice una palabra y el niño debe pulsar la tecla correspondiente a la misma



Talking Calculator
Es uno de los programas más interesantes de los que dispone My Talking Computer. Pulsando los números y los signos aritméticos se pueden calcular sumas sencillas



Decir la hora
El ordenador lleva instalado My Talking Clock (mi reloj parlante) encima del teclado al tacto. En la cara inferior del reloj hay un conjunto de protuberancias, que corresponden al teclado de los ordenadores. Las manecillas del reloj se pueden mover alrededor del cuadrante y el micro dirá qué hora se está visualizando

(ian McKinnell)

La máquina le pide entonces al niño que pulse la tecla GO de la cubierta. El ordenador "sabe" cuál es el programa que se ha seleccionado porque las teclas ON y GO están situadas en distintas posiciones en cada cubierta. Este método de operación es especialmente útil para los niños de corta edad.

Una vez que el niño ha aprendido que ON hace que el ordenador se encienda y que GO da comienzo al programa, puede luego cargar y ejecutar cualquiera de los programas sin que para ello se requiera la ayuda de un adulto. Las cubiertas están revestidas en plástico, lo que significa que las eventuales manchas de mermelada y chocolate se podrán limpiar.

Para evaluar el valor educativo de un elemento de hardware o de software se han de aplicar ciertos criterios. El paquete debe enseñar aquello que pretende, por ejemplo, y no debe presuponer ningún conocimiento previo del tema que intenta impartir: las páginas de instrucciones escritas son evidentemente inaceptables en un programa para "aprender a leer". También sirve de ayuda el hecho de que sea entretenido y fácil de utilizar.

El sintetizador de voz

El sintetizador de voz incorporado es lo mejor del ordenador. De hecho, hace que uno se pregunte por qué las calculadoras no poseen facilidades de voz incorporadas. Así y todo, tiene sus limitaciones. El habla está almacenada como palabras enteras, a las que se accede luego individualmente para construir las frases utilizadas. La enfática pronunciación resultante en realidad no se puede evitar, puesto que las mismas palabras se aplican en numerosos contextos. La voz femenina utilizada posee acento norteamericano, lo que tal vez induzca a que los padres británicos se quejen de que se les enseñe a sus hijos a hablar en "norteamericano", pero ello molestará más a los adultos que a los niños. El problema más grave, tratándose de una máquina diseñada para aplicaciones educativas, es que muchos de los vocablos no son suficientemente claros: es difícil, por ejemplo, diferenciar las palabras *"by"*, *"my"* y *"sky"*. Incluso tras la repetición de ciertas frases resulta difícil determinar lo que se ha dicho.

Ahora podemos juzgar si My Talking Computer satisface los criterios que hemos definido como esenciales en todo medio auxiliar educativo. La facilidad con la cual se pueden instalar y ejecutar nuevos programas es ciertamente digna de elogio. El hecho de que el ordenador hable es una enorme ventaja para enseñar a un niño a leer, y evita hábilmente la necesidad de complicadas páginas de instrucciones. A pesar de las dudas sobre la calidad de la voz, la capacidad del ordenador para asociar el sonido de la palabra con su forma escrita es un método de enseñanza mucho más directo que el que ofrecen los paquetes de software, que simplemente asocian la palabra con imágenes; aunque, por supuesto, My Talking Computer también lo hace. Su capacidad para responder verbalmente asegura, asimismo, que su utilización sea entretenida.

My Talking Computer no es, como es obvio, un ordenador en el sentido literal de la palabra, puesto que el usuario no lo puede "programar". Sin embargo, demuestra la aplicación de la tecnología del microordenador en el campo de la educación.

MY TALKING COMPUTER

DIMENSIONES
250×230×70 mm

SOFTWARE
El software se proporciona en la ROM del ordenador. Se ofrece software extra en forma del Expansion Module 1, que se enchufa en una ranura para cartuchos que posee la máquina

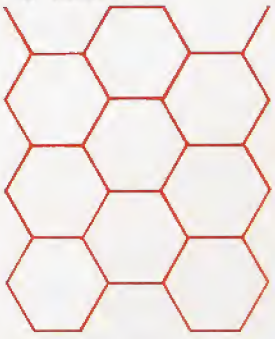
VENTAJAS
Es sumamente fácil de utilizar. La sencillez del procedimiento de carga y la facilidad para síntesis de voz significan que el niño no necesita saber leer para poder utilizar la máquina

DESVENTAJAS
Con frecuencia el habla es enfática y poco clara. La máquina no es programable y no se puede considerar un auténtico microordenador

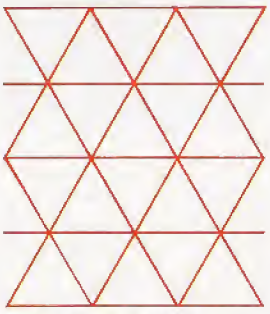
Líneas finales

En este último capítulo de nuestro curso veremos cómo se utiliza el LOGO para teselar diversas y atractivas formas geométricas

Tres teselados



Utilización de mosaicos hexagonales



Utilización de mosaicos de triángulo equilátero

Utilización de mosaicos cuadrados

Liz Dixon

Un *teselado* cubre la superficie de un plano mediante la unión de mosaicos de la misma forma, sin dejar espacios entre las piezas individuales.

Algunos polígonos regulares formarán teselados: cuadrados, triángulos equiláteros y hexágonos (como vemos en el margen izquierdo). Sin embargo, ninguno de los otros polígonos se puede teselar. Para ver por qué, tomemos un ejemplo: un pentágono. El ángulo del vértice de un pentágono regular es de 108°. Coloque tres de ellos juntos en un punto focal y habrá empleado sólo 324°, de modo que queda un vacío. Añada un cuarto y el ángulo será entonces de 432°, lo que significa que las formas se superponen. Tanto los cuadrados como los triángulos equiláteros y los hexágonos se teselan precisamente porque los ángulos de sus vértices (90°, 60° y 120° respectivamente) dividen 360 un número exacto de veces.

Escribir procedimientos en LOGO para producir estos teselados es bastante directo. Probablemente el mejor enfoque sea escribir un procedimiento para dibujar el motivo empezando desde el centro y retornando al centro (ello supone una transparencia de estado). Luego podemos utilizarlo dentro de un "superprocedimiento" que desplace la tortuga hasta el centro de la siguiente forma a dibujar en cada ocasión.

Los otros procedimientos dibujan un teselado simple de cuadrados. Los teselados de triángulos equiláteros y hexágonos se pueden dibujar de forma similar.

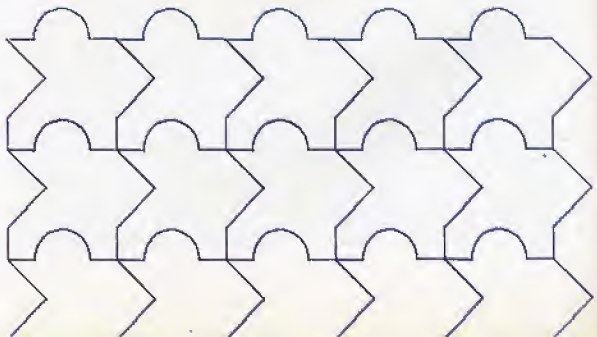
```
TO TES
  PENUP SETXY (-100)(90) PENDOWN
  REPEAT 5[LINEAVERT SETXY (-100)
  (YCOR-40)
END

TO LINEAVERT
  REPEAT 5[CUAD 40 SETXY XCOR+40]
END

TO CUAD :S
  PENUP LEFT 45
  FORWARD :S*(SQRT 2)/2
  RIGHT 135 PENDOWN
  FORWARD :S RIGHT 90 FORWARD :S RIGHT 90
  FORWARD :S RIGHT 90 FORWARD :S RIGHT 90
  PENUP LEFT 135
  BACK :S*(SQRT2)/2 RIGHT 45
END
```

Se pueden componer teselados a partir de motivos más complejos que los polígonos regulares y también se pueden realizar utilizando una mezcla de formas. No obstante, los teselados basados en polígonos regulares poseen un inmenso potencial. Muchos de los dibujos de M.C. Escher son variaciones de teselados regulares simples.

Un método directo para construir diagramas más complejos implica modificar CUADRADO reemplazando las instrucciones FORWARD que dibujan los lados del cuadrado por procedimientos. La única regla que debemos observar es que cualquier modificación en el lado superior del cuadrado se debe compensar con otra en la base, y cualquier modificación en el lado derecho se ha de compensar con otra en el lado izquierdo, de modo que las formas sigan encajando.

Para dibujar la forma básica, sencillamente reemplazamos las instrucciones que dibujan los lados del cuadrado por procedimientos para dibujar las líneas nuevas. El nuevo teselado (que vemos abajo) se dibuja entonces volviendo a llamar a TES. Éste es el listado completo:

```
TO CUAD :S
  PENUP LEFT 45
  FORWARD :S*(SQRT 2)/2
  RIGHT 135 PENDOWN
  LADOA :S RIGHT 90 LADOB :S
  RIGHT 90
  LADOC :S RIGHT 90 LADOD :S
  RIGHT 90
  PENUP LEFT 135
  BACK :S*(SQRT 2)/2
  RIGHT 45
END

TO LADOA :S
  FORWARD :S/4 LEFT 90
  REPEAT 19[FORWARD 2*3.1416* :S/144
  RIGHT 10]
  LEFT 100 FORWARD :S/4
END

TO LADOB :S
  LEFT 45 FORWARD :S/2
  RIGHT 90 FORWARD :S/2
  LEFT 45 FORWARD :S-:S*(SQRT 2)/2
END

TO LADOC :S
  FORWARD :S/4 RIGHT 90
  REPEAT 19 [FORWARD 2*3.1416*:S/144 LEFT10]
  RIGHT 100 FORWARD :S/4
END
```



```
TO LADOD :S
  FORWARD :S-:S*(SQRT 2)/2 RIGHT 45
  FORWARD :S/2 LEFT 90
  FORWARD :S/2 RIGHT 45
END
```

La técnica de Escher era, por supuesto, más complicada, puesto que él también modificaba lentamente las formas a medida que se iban "moviendo" a través de su teselado. Sería interesante un procedimiento en LOGO que hiciera lo mismo...

Usando pegamento

Una forma bastante diferente de enfocar la cuestión de los teselados es la que ofrecen Harold Abelson y Andrea diSessa a modo de ejercicio en su libro *Turtle geometry*. Su enfoque es así: imagine una forma dibujada en el interior de un cuadrado, por ejemplo, el patrón de un mosaico, y después pegue cuatro de éstos entre sí para formar un cuadrado más grande. Tome este cuadrado más grande y pegue cuatro de éstos entre sí, y así sucesivamente.

Existen muchos métodos diferentes de pegar. De hecho, puesto que cada una de las cuatro piezas se puede orientar en una de cuatro direcciones, hay 4×4×4×4, o 256, posibilidades. Los patrones resultantes son producto tanto de lo que se había dibujado en el mosaico como del método de pegado empleado. Nuestro procedimiento, MARCA, dibuja el motivo básico que vemos a la derecha.

```
TO MARCA :S
  PENDOWN LEFT 45
  FORWARD :S*(SQRT 2)/2
  BACK :S*(SQRT 2)/2
  LEFT 90 FORWARD :S*(SQRT 2)/2 PENUP
  BACK :S*(SQRT 2) PENDOWN
  FORWARD :S*(SQRT 2)/4 LEFT 45
  FORWARD :S/2 LEFT 45
  FORWARD :S*(SQRT 2)/4 PENUP
  BACK :S*(SQRT 2)/2 LEFT 135
END
```

Ian McKinnell

Figuras en evolución
Las figuras de teselado son formas que encajan entre sí exactamente para cubrir un plano. *Metamorfosis II*, de M. C. Escher (1898-1972), en el cual está basado nuestro gráfico creado por ordenador, demuestra cómo una sencilla figura teselada, como un hexágono regular, se puede ir modificando progresivamente para producir un teselado más complejo; en este caso, la forma de una sofisticada lagartija. La técnica de Escher pone de manifiesto el cambio paulatino de lo simple a lo complejo a medida que el ojo va avanzando por la escena

Un posible procedimiento de pegado sería PEGAR1. Para ejecutarlo, digite DRAW PENUP PEGAR1 [MARCA]100. Podemos ver el patrón resultante.

```
TO PEGAR1 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
END
```

PIEZA.CUADRADO dibuja uno de los cuatro cuadrados que componen el cuadrado más grande. Toma tres entradas: :A determina la orientación del mismo, :PROC es el nombre del procedimiento para dibujar el motivo y :S es la longitud del lado. No es factible, dados los límites de la pantalla, mantener en continua ampliación el tamaño de los lados, de modo que el procedimiento toma el tamaño del cuadrado exterior como entrada y calcula el tamaño de los cuadrados más pequeños.

```
TO PIEZA.CUADRADO :A :PROC :S
  FORWARD :S / 4 RIGHT 90
  FORWARD :S / 4 RIGHT 90*:A
  RUN SENTENCE :PROC :S / 2 LEFT 90*:A
  BACK :S / 4 LEFT 90
  BACK :S / 4 RIGHT 90
END
```

He aquí algunos otros posibles métodos de pegado:

```
TO PEGAR2 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
  PIEZA.CUADRADO 1 :PROC :S
  PIEZA.CUADRADO 2 :PROC :S
  PIEZA.CUADRADO 3 :PROC :S
END

TO PEGAR3 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
  PIEZA.CUADRADO 2 :PROC :S
  PIEZA.CUADRADO 3 :PROC :S
  PIEZA.CUADRADO 1 :PROC :S
END

TO PEGAR4 :PROC :S
  PIEZA.CUADRADO 3 :PROC :S
  PIEZA.CUADRADO 2 :PROC :S
  PIEZA.CUADRADO 1 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
END

TO PEGAR5 :PROC :S
  PIEZA.CUADRADO 2 :PROC :S
  PIEZA.CUADRADO 1 :PROC :S
  PIEZA.CUADRADO 0 :PROC :S
  PIEZA.CUADRADO 3 :PROC :S
END
```

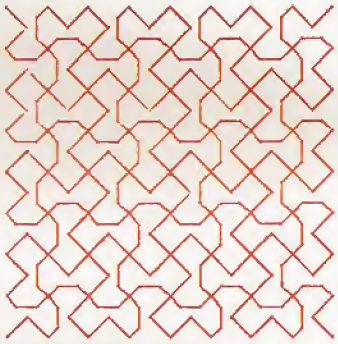
MARCA es el nombre de la instrucción para dibujar la forma que requiere una entrada. Esta la pegamos con PEGAR1 [MARCA] 100. Observando esta línea de instrucciones podemos ver que PEGAR1 [MARCA] se puede considerar como una instrucción que sólo exige un número con el objeto de dibujar una forma. Podemos, por lo tanto, utilizar [PEGAR1 [MARCA]] como la entrada para otra instrucción PEGAR. Por ejemplo:

```
PEGAR2 [PEGAR1 [MARCA]] 100
```

¡La diversión apenas empieza! Considere:

```
PEGAR3 [PEGAR2[PEGAR1[MARCA]]]100
```

Ahora tenemos 256×256×256 combinaciones de pegado a tres niveles para considerar. ¡Pero tampoco hay razón para detenerse en tres niveles.

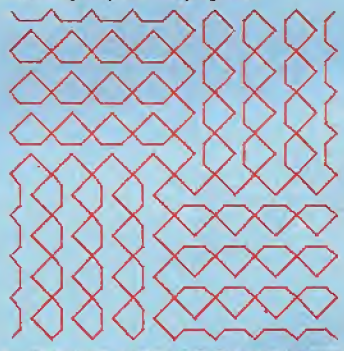


Complementos al LOGO

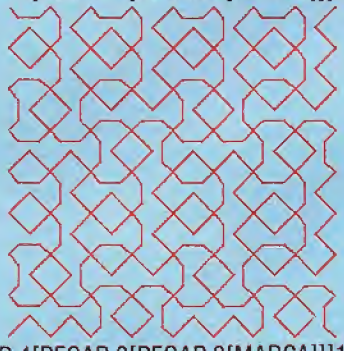
En todas las versiones LCSI, utilice SETPOS en vez de SETXY, y recuerde que ha de ir seguida de una lista. En el LOGO Atari debe emplear la abreviatura SE en vez de SENTENCE

¿Sin ideas?

He aquí otros ejemplos de pegado:



PEGAR 1[PEGAR 4[PEGAR 5[MARCA]]]100



PEGAR 4[PEGAR 3[PEGAR 3[MARCA]]]100

Misión de reconocimiento

Diseñaremos software para que el robot explore y visualice una superficie determinada

Para lograr que nuestro robot explore un área dada, debemos primero concebir un patrón de exploración que cubra la zona. Podríamos empezar por considerar un patrón de exploración en el cual el robot se mueva hacia atrás y hacia adelante a través de la superficie sondeando en busca de algún objeto. Al localizar lo podría optar por tantear alrededor de sus lados antes de proceder a "barrer" el resto de la superficie. Sin embargo, esta clase de algoritmo es difícil de programar y puede plantear problemas si dentro de la zona a explorar hay más de un objeto. Un curso de acción alternativo al encontrar el objeto es dar un rodeo y continuar explorando como antes. Los pasos de un algoritmo de esta clase se pueden describir así:

```
REPEAT
  REPEAT
  Sondear hacia adelante en busca de un objeto
  UNTIL llegar al borde del área definida
  o hallar un objeto
Avanzar un "ancho de franja"
Girar
UNTIL llegar al borde del área definida
```

Al completar esta exploración horizontal, el robot no habrá sondeado toda el área disponible si en la misma hubiera algún objeto presente. Utilizando este patrón de exploración, la superficie por detrás de cualquier objeto ("la zona ciega") quedará sin explorar. Se requiere al menos una exploración vertical (es decir, a 90° de la primera).

Para que el robot pueda llevar a cabo una exploración satisfactoria es necesario introducir una mínima modificación en los dos sensores delanteros. Debido a que las ruedas del robot sobresalen por los lados, existe el peligro de que entren en contacto con un objeto mientras el robot pasa explorando junto a su lado. Para subsanar este inconveniente hemos de añadir un escudo adelante de cada sensor para proteger las ruedas. Cada escudo ha de ser un rectángulo de aproximadamente 95×25 mm, de un material ligero (plástico duro o metal). Asegúrese de que el material empleado no sea tan pesado como para empujar el sensor hacia abajo cuando esté en posición cerrado. Los escudos se deben montar encima de los sensores utilizando cinta adhesiva o cola, y se deben colocar de modo que se extiendan hacia afuera para cubrir exactamente las ruedas. Compruebe la acción del interruptor del sensor para asegurar que se pueda abrir y cerrar.

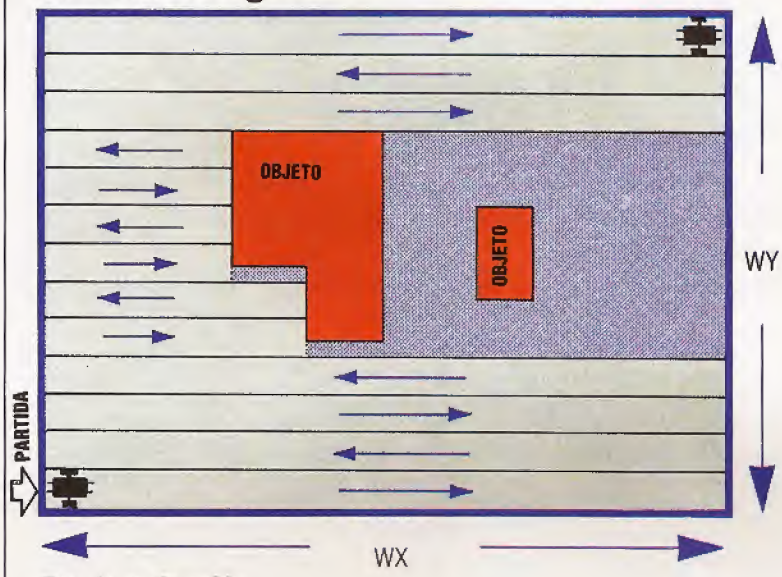
El programa utilizado para hacer que el robot efectúe una exploración de superficie se ha escrito empleando un único patrón de exploración horizontal y vertical. A medida que el robot va completando su recorrido de cada franja de la exploración, se va rellenando con color la zona correspondiente en la pantalla del ordenador. Finalizada la exploración, se visualizará en la pantalla una representación pictórica de la superficie, tal como la "ve" el robot.

Las dos versiones de nuestro programa piden al usuario que entre las dimensiones de la superficie a explorar. Utilizando la Mode 4 del BBC o la visualización en alta resolución del Commodore 64, se hace que cada pixel de pantalla corresponda a un cuadrado de 4 mm de la superficie que se esté explorando. Las dimensiones máximas en las direcciones horizontal y vertical son, por consiguiente, 1 279 y 1 023 mm, respectivamente, para ambas máquinas. Observe que cada programa utiliza un ancho de franja de 40 mm.

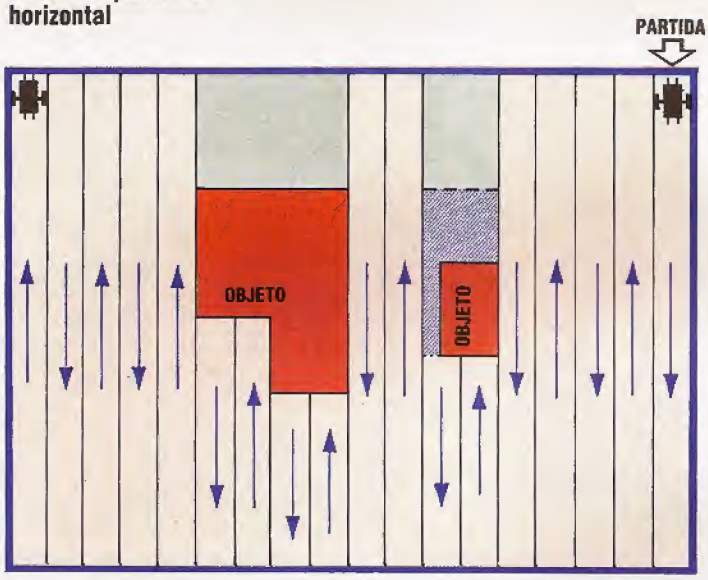
A medida que el robot, durante la exploración, va completando cada una de las franjas, la visualización gráfica se va actualizando utilizando los procedimientos (o subrutinas) denominados XPLOT e YPLOT. En la versión para el BBC Micro, la superficie de franjas de la pantalla se rellena empleando las instrucciones MOVE y DRAW para producir una serie de líneas paralelas adyacentes. La versión para el Commodore 64, sin embargo, se basa en las dos rutinas en código máquina (PLOTSUB y LINESUB) diseñadas previamente en el curso para compensar la falta de instrucciones para dibujar en alta resolución del BASIC Commodore. Usted debe de tener copia de los archivos objeto finales para estas dos rutinas, que se pueden cargar utilizando las líneas 30 y 40 de la versión Commodore del programa. Por otra parte, tendrá cargadores en BASIC para estas dos rutinas. Cada rutina se debe cargar y ejecutar por separado antes de cargar y ejecutar el programa. En este último caso, las líneas 30 y 40 se pueden omitir.

Además de la zona ciega sustancial que podría quedar tras dos exploraciones, se pueden presentar otras dificultades adicionales. Entre ellas, la más destacable es que el programa trabaja con niveles de éxito variables para formas u orientaciones diferentes. Tendrá el mayor éxito con formas rectangulares, orientadas a lo largo de los ejes horizontal y vertical. Las triangulares y circulares producirán zonas ciegas más grandes, dado que el patrón de exploración tenderá a simplificar las irregularidades de los objetos que se encuentren. Quizá desee, a modo de ejercicio, ampliar el programa que ofrecemos aquí de modo que se efectúe otra exploración vertical. El robot puede, asimismo, hallarse en dificultades si se encuentra con un objeto durante su maniobra para comenzar una franja nueva.

Puntos ciegos



Tras la exploración horizontal



Tras la exploración vertical

Clave:

Sup. explorada horizontalmente

Sup. explorada verticalmente

Sup. sin explorar

El método más simple para explorar una superficie rectangular dada consiste en efectuar dos "barridos" a 90° uno del otro. Cada vez que se encuentre un objeto, o cuando se llegue al borde de la superficie, el robot avanza una franja hacia arriba y continúa el barrido en la dirección contraria.
Aplicando este procedimiento, detrás de cada objeto hallado quedarán "zonas ciegas". El segundo barrido, sin embargo, proporcionará gran parte de los detalles que pasaron "desapercibidos" durante el primer barrido. Aun así se producirán imprecisiones, debido al ancho de la franja de barrido utilizada y la orientación mutua de los objetos

Kevin Jones

Listado para el BBC Micro

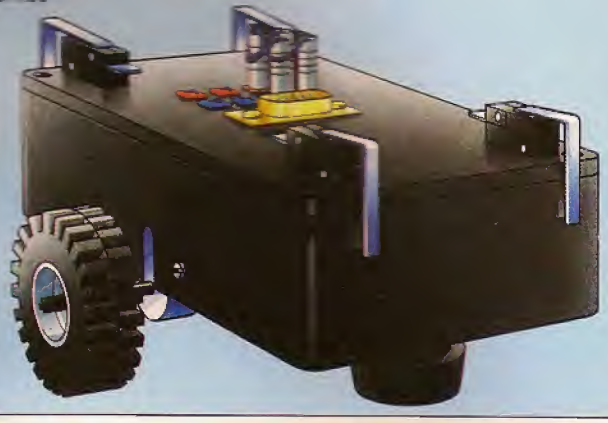
```
10 REM **** EXPLORADOR DE FORMAS BBC ****
20 MODE 4
30 PROCinicializar
40 PROCexplorar__dimensiones
50 PROCexplor__horiz
60 PROCexplor__vert
70 END
80 DEF PROCexplor__horiz
90 objetivo=wx:sentido=izquierda:xorigen=0
100 REPEAT
110 REPEAT
120 PROCmover(adelante,dx):x=x+dx
130 UNTIL(?REGDAT AND 192)<>ningun__parachoques OR x=objetivo
140 PROCxplot
150 dx=-dx:y=y+dy:xorigen=x
160 IF objetivo=0 THEN objectivo=wx ELSE objetivo=0
170 IF y<wy THEN PROCsig__franja(sentido)
180 IF sentido=derecha THEN sentido=izquierda ELSE sentido=derecha
190 UNTIL y>wy
200 ENDPROC
210 :
220 DEF PROCexplor__vert
230 IF x=0 THEN sentido=izquierda:dx=anchura ELSE sentido=derecha:dx=-anchura
```

Listado para el Commodore 64

```
10 REM **** EXPLORADOR DE FORMAS CBM ****
20 DN=8:REM SI CASS DN=1
30 IF A=0 THEN A=1:LOAD"PLOTSUB.HEX",DN,1
40 IF A=1 THEN A=2:LOAD"LINESUB.HEX",DN,1
50 GOSUB1000:REM INICIALIZAR
60 GOSUB2000:REM EXPLORAR DIMENSIONES
70 GOSUB5000:REM PASAR A MODALIDAD ALTA RES
80 GOSUB3000:REM EXPLORACION HORIZONTAL
90 GOSUB4000:REM EXPLORACION VERTICAL
100 GOSUB5100:REM SALIR DE MODALIDAD ALTA RES
110 END
120 :
1000 REM **** S/R INICIALIZAR ****
1010 RDD=56579:REGDAT=56577
1020 POKE RDD,15:POKE REGDAT,1
1030 AD=4:AT=2:IZ=6:DE=0
1040 ID=3.34446:IA=375/90
1050 RB=128:LB=64:BB=0:NB=192
1060 WD=40:DW=WD/10:X=0:Y=0:DX=DW:DY=WD
1070 REM ** DIRECCIONES DE COMIENZO C/M **
1080 HIRES=49422:LINESUB=49934
1090 RETURN
1100 :
```

Continuación del listado para el BBC Micro

```
240 dy=-dw:objetivo=0:yorigen=y
250 PROCgirar(sentido,90)
260 REPEAT
270 REPEAT
280 PROCmover(adelante,dy):y=y+dy
290 UNTIL(?REGDAT AND 192)<>ningun_parachoques OR y=objetivo
300 PROCyplot
310 dy=-dy:x=x+dx:yorigen=y
320 IF objetivo=0 THEN objetivo=wy ELSE objetivo=0
330 PROCsig_franja(sentido)
340 IF sentido=derecha THEN sentido=izquierda ELSE sentido=derecha
350 UNTIL x>wx OR x<0
360 ENDPROC
370 :
380 DEF PROCsig_franja(rumbo)
390 PROCmover(atras,30)
400 PROCgirar(rumbo,90)
410 PROCmover(adelante,anchura)
420 PROCgirar(rumbo,90)
430 ENDPROC
440 :
450 DEF PROCexplorar_dimensiones
460 INPUT"DIMENSION X EN MM";wx
470 IF wx>1279 THEN 460
480 INPUT"DIMENSION Y EN MM";wy
490 IF wy>1023 THEN 480
500 wx=anchura*(wx DIV anchura)
510 wy=anchura*(wy DIV anchura)
520 CLS
530 ENDPROC
540 :
550 DEF PROCinicializar
560 FDD=&FE62:REGDAT=&FE60
570 ?FDD=15 REM LINEAS 0-3 SALIDA
580 ?REGDAT=1 REM ENCENDER BIT REINICIO
590 adelante=4:atras=8:izquierda=5:derecha=9
600 coef_id=3.3:coef_ia=375/90
610 parachoques_derecho=128:parachoques_izquierdo=64
620 ambos_parachoques=0:ningun_parachoques=192
630 anchura=40:dx=anchura*8
640 x=0:y=0:dx=0:dy=anchura
650 MOVE 0,0
660 ENDPROC
670 :
680 DEF PROCmover(dir,distancia)
690 ?REGDAT=(?REGDAT AND 1)OR dir
700 impulsos=coef_id*ABS(distancia)
710 FOR I=1 TO impulsos:PROCimpulso:NEXT I
720 ENDPROC
730 :
740 DEF PROCgirar(dir,angulo)
750 ?REGDAT=(?REGDAT AND 1)OR dir
760 impulsos=coef_ia*angulo
770 FOR I=1 TO impulsos:PROCimpulso:NEXT I
780 ENDPROC
790 :
800 DEF PROCimpulso
810 ?REGDAT=(?REGDAT OR 8)
820 ?REGDAT=(?REGDAT AND 247)
830 ENDPROC
840 DEF PROCxplot
850 FOR I=0 TO anchura STEP 4
860 MOVE xorigen,y+I
870 DRAW x,y+I
880 NEXT I
890 ENDPROC
900 :
910 DEF PROCyplot
920 FOR I=0 TO anchura STEP 4
930 MOVE x+I,yorigen
940 DRAW x+I,y
950 NEXT I
960 ENDPROC
```



Continuación del listado para el Commodore 64

```
2000 REM **** DIMENSIONES DE EXPLORACION ****
2010 INPUT"DIMENSION X EN MM";WX
2020 IF WX/4>319 THEN 2010
2030 INPUT"DIMENSION Y EN MM";WY
2040 IF WY/4>199 THEN 2030
2050 WX=WD*INT(WX/WD):WY=WD*INT(WY/WD)
2060 RETURN
2070 :
3000 REM **** EXPLORACION HORIZONTAL ****
3010 TG=WX:SE=LF:XS=0
3020 DR=FW:DS=DX:GOSUB7000:REM MOVER
3030 X=X+DX
3040 IF(PEEK(REGDAT)AND 192)=NB AND X<>TG THEN 3020
3050 GOSUB 9000:REM XPLOT
3060 DX=-DX:Y=Y+DY:XS=S
3070 IF TG=0 THEN TG=WX:GOTO 3090
3080 TG=0
3090 IF Y<WY THEN WA=SE:GOSUB8000:REM SIGUIENTE FRANJA
3100 IF SE=RT THEN SE=LF:GOTO 3120
3110 SE=RT
3120 IF Y<WY THEN 3020
3130 RETURN
3140 :
4000 REM **** EXPLORACION VERTICAL ****
4010 IF X=0 THEN SE=LF:DX=WD:GOTO 4030
4020 SE=RT:DX=-WD
4030 DY=-DY:TG=0:YS=Y
4040 DR=SE:AG=90:GOSUB7100:REM GIRAR
4050 DR=FW:DS=DY:GOSUB7000:REM MOVER
4060 Y=Y+DY
4070 IF(PEEK(REGDAT)AND192)=NB AND Y<>TG THEN 4050
4080 GOSUB9100:REM YPLOT
4090 DY=-DY:X=X+DX:YS=Y
4100 IF TG=0 THEN TG=WY:GOTO 4120
4110 TG=0
4120 WA=SE:GOSUB8000:REM SIGUIENTE FRANJA
4130 IF SE=RT THEN SE=LF:GOTO 4150
4140 SE=RT
4150 IF X<WX AND X>0 THEN 4050:REM REPETIR
4160 RETURN
4170 :
5000 REM **** ENTRAR ALTA RES ****
5010 POKE 49408,1:POKE 49409,1
5020 POKE 49410,1:SYS HIRES:RETURN
5030 :
5100 REM **** SALIR DE ALTA RES ****
5110 POKE 49408,0:POKE 49409,0
5120 POKE 49410,1:SYS HIRES:RETURN
5130 :
6000 REM **** ENTRAR LINESUB ****
6010 MHI=INT(X1/256):MLO=X1-256*MHI
6020 NHI=INT(X2/256):NLO=X2-256*NHI
6030 POKE 49920,MLO:POKE 49921,MHI
6040 POKE 49922,NLO:POKE 49923,NHI
6050 POKE 49924,Y1:POKE 49925,Y2
6060 SYS LINESUB:RETURN
6070 :
7000 REM **** MOVER (DR,DS) ****
7010 POKE REGDAT,(PEEK(REGDAT)AND 1)OR DR
7020 PL=PD*DS
7030 FOR I=1 TO PL:GOSUB7200:NEXT I
7040 RETURN
7050 :
7100 REM **** GIRAR (DR,AG) ****
7110 POKE REGDAT,(PEEK(REGDAT)AND 1)OR DR
7120 PL=PA*DS
7130 FOR I=1 TO PL:GOSUB7200:NEXT I
7140 RETURN
7150 :
7200 REM **** IMPULSO ****
7210 POKE REGDAT,PEEK(REGDAT)OR 8
7220 POKE REGDAT,PEEK(REGDAT)AND 247
7230 RETURN
7240 :
8000 REM **** SIGUIENTE FRANJA ****
8010 DR=BW:DS=30:GOSUB7000:REM MOVER
8020 DR=WA:AG=90:GOSUB7100:REM GIRAR
8030 DR=FW:DS=WD:GOSUB7000:REM MOVER
8040 DR=WA:AG=90:GOSUB7100:REM GIRAR
8050 RETURN
9000 REM **** XPLOT ****
9010 FOR I=0 TO WD
9020 X1=XS/4:Y1=(Y+I)/4:X2=X/4:Y2=Y1
9030 GOSUB 6000:REM ENTRAR LINESUB
9035 NEXT I
9040 RETURN
9100 REM **** YPLOT ****
9110 FOR I=0 TO WD
9120 X1=(X+I)/4:Y1=YS/4:X2=X1:Y2=Y/4
9130 GOSUB6000:REM ENTRAR LINESUB
9135 NEXT I
9140 RETURN
```

Punto de partida

Una vez estudiadas las llamadas OSBYTE, nos corresponde analizar las características de otro grupo de llamadas igualmente útiles: las OSWORD

Las llamadas OSBYTE tienen una limitación importante: sólo aceptan dos parámetros, uno en X y otro en Y. Contra lo cual nada hay que objetar siempre y cuando no se desee pasar un buen número de parámetros al sistema operativo, aunque en realidad, y aun poniéndole mucha voluntad, difícilmente se pueden controlar todas las rutinas con tan sólo dos parámetros. Por su parte, las rutinas OSWORD son capaces de controlar más de dos parámetros y resultan una utilísima alternativa a las llamadas OSBYTE.

OSWORD emplea un área de memoria conocida por *bloque de parámetros* para el paso de éstos al sistema operativo. Esta área, de tamaño variable según la llamada OSWORD que se está utilizando, se puede situar en cualquier lugar dentro de la RAM del usuario. Cuando se llama una rutina OSWORD, quien se encarga de decir cuál de las rutinas es la llamada es el registro A, mientras que los registros X e Y especifican la dirección del primer byte del bloque de parámetros. El registro X retiene el byte inferior de dicha dirección y el registro Y el superior. Por ejemplo, si el bloque se sitúa en la dirección &0A00, el registro X retiene 00 y el Y el valor &0A:

Ensamblador	BASIC
LDX #0	X%=0
LDY #&0A	Y%=&0A

Se produce la llamada OSWORD en la dirección &FFF1, que está vectorizada del mismo modo que una llamada OSBYTE. Pero el vector OSWORD se encuentra en la dirección &20C. Por tanto, para llamar una rutina OSWORD, especificada con el registro A conteniendo el valor 1, estableceremos el bloque con los parámetros necesarios, y después llamaremos la rutina por medio de las siguientes líneas en código (parameter_block es la dirección del primer byte del bloque):

```
LDX   # parameter_block MOD 256
LDY   # parameter_block DIV 256
LDA   #1
JSR   &FFF1
```

Naturalmente los valores del bloque de parámetros dependerán de la llamada concreta que se utilice.

¿Qué hacen las llamadas OSWORD?

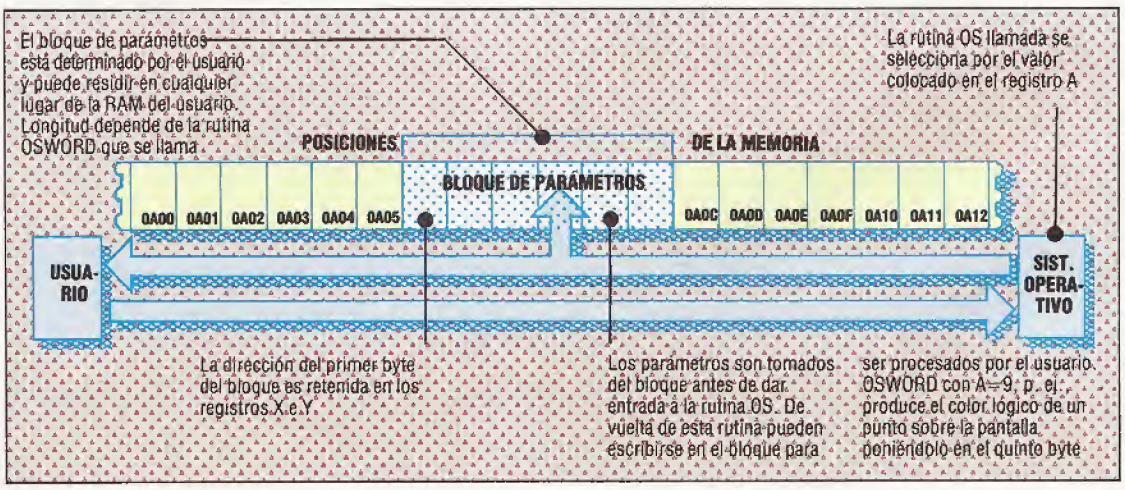
Con estas llamadas podemos leer información de la corriente actual de entrada y pasarla a la memoria, ejecutar instrucciones SOUND y ENVELOPE, acceder al sistema de ficheros en disco, y otras muchas funciones.

Veamos ahora cómo se utilizan algunas de estas llamadas OSWORD. Para identificarlas, se nombran según el contenido de A cuando se realiza la llamada a la dirección &FFF1: por ejemplo, OSWORD con A=0.

• *OSWORD con A=0*: Esta llamada nos permite leer una línea de entrada según la corriente seleccionada de entrada, lo que constituye una función de máxima utilidad. Nos permite especificar el número máximo de caracteres que serán aceptados y los valores máximo y mínimo permitidos en los códigos ASCII de los caracteres que han de entrar. Los datos leídos se almacenan en la memoria en la dirección especificada en los primeros dos bytes del bloque de parámetros.

Este bloque de la llamada OSWORD se establece de la siguiente manera:

Más pista
Una llamada OSWORD proporciona el medio más conveniente para controlar las rutinas del OS que necesitan más de los dos parámetros disponibles en las rutinas OSBYTE



Liz Dixon

Bloque de parámetros para OSWORD con A=0	
Byte	Función
0	Byte inferior de la dirección del buffer
1	Byte superior de la dirección del buffer
2	Número máximo de caracteres en la entrada
3	Máximo código ASCII aceptable
4	Mínimo código ASCII aceptable

Los registros X e Y señalan al byte 0 del bloque. Este programa escrito a continuación ilustra el uso de la llamada; el bloque se establece de tal manera que se puedan aceptar hasta 7 caracteres en código ASCII que vayan del 32 al 90, ambos inclusive.

```
 10 DIM C20
 20 A%=0
 30 X%=C MOD 256
 40 Y%=C DIV 256
 50 C?0=&00
 60 C?1=&0A:REM buffer en &0A00
 70 C?2=7
 80 C?3=32
 90 C?4=90
100 CALL &FFF1
110 PRINT $(&0A00)
```

Pulse RUN para ejecutarlo y digite algunos caracteres. La tecla Delete funcionará como siempre, y si pulsa Return o Escape provocará el final de esta rutina OSWORD. Es importante observar que si se intenta digitar caracteres cuyos códigos ASCII caen fuera del intervalo especificado, serán escritos en la pantalla pero no se agregarán a los caracteres escritos en el buffer. Si usted escribe demasiados caracteres, lo que oirá será un "pitido" de aviso. En lenguaje máquina, cuando se sale de la llamada OSWORD, el flag C se activa en caso de pulsar la tecla Escape; si esto no es así, C estará a 0. El registro Y retendrá el número de caracteres introducidos, incluyendo posibles retornos de carro.

- *OSWORD con A=1 y A=2*: Estas llamadas permiten al programador en código máquina acceder a la variable TIME del BASIC. La llamada OSWORD con A=1 nos permite leer el valor actual de TIME. Los cinco bytes que representan este valor se almacenan en el bloque de parámetros, que, obviamente, habrá de tener una longitud de cinco bytes para esta llamada. El byte menos significativo del valor es almacenado en el byte 0 del bloque y el más significativo en el byte 5.

OSWORD con A=2 nos permite dar a TIME un valor determinado. El byte menos significativo se coloca en el primer byte del bloque, el más significativo en el quinto, y seguidamente se realiza la llamada a OSWORD. El siguiente programa da a TIME el valor 100:

```
 10 DIM C 20
 20 X%=C MOD 256
 30 Y%=C DIV 256
 40 A%=2
 50 C?0=100
 60 C?1=0
 70 C?2=0
 80 C?3=0
 90 C?4=0
100 CALL &FFF1
```

- *OSWORD con A=3 y A=4*: El BBC posee un segundo reloj interno llamado *interval timer*, o bien *event timer*. No es empleado por TIME pero sí se utiliza por el *sistema de eventos* del ordenador, del que hablaremos más adelante. Este cronómetro comienza a contar, a la velocidad de un tic-tac cada centésima de segundo, partiendo de un determinado valor y genera un "evento" cuando llega a cero. Este evento se puede manipular fácilmente y puede servir para que el ordenador dé comienzo a una serie de operaciones transcurrido un tiempo determinado.

OSWORD con A=3 nos permite leer el valor actual del cronómetro *interval timer*, transfiriéndolo al bloque de parámetros, de un modo semejante a como OSWORD con A=1 leía el reloj de TIME. Para dar un determinado valor a este cronómetro se emplea OSWORD con A=4, así como con OSWORD con A=2 se daba un valor a TIME. Si, por ejemplo, hacemos que el bloque de parámetros retenga &FFFFFFFC, el evento ha de producirse transcurridos 0,004 segundos (4 milésimas de segundo).

- *OSWORD con A=5 y A=6*: Se trata de dos llamadas muy especiales, de extrema utilidad para aquellos programadores que tengan acceso al segundo procesador del BBC Micro. Una vez acoplado al ordenador uno de estos dispositivos, se comunicará a través de la interface Tube y considerará al ordenador como un procesador de E/S. Lo que significa que el BBC Micro actúa ahora como un dispositivo esclavo, realizando todas las tareas de servicio que le ordena el segundo procesador, como tratar las entradas del teclado, visualizar en pantalla, y toda función genérica de entrada/salida. Estas llamadas OSWORD permiten al segundo procesador el acceso a las posiciones de memoria del procesador de E/S.

Bloques de parámetros para OSWORD con A=5 y A=6		
Byte	OSWORD con A=5	OSWORD con A=6
0	BmnS de la dir. del proces. E/S	BmnS de la dir. del proces. E/S
1	Dirección del byte por leer	Dirección del byte por escribir
2	Dirección del byte por leer	Dirección del byte por escribir
3	BmsS del procesador E/S	BmsS del procesador E/S
4	Tras la llamada, es el byte contenido en la dirección superior	Byte que hay que escribir en la dirección superior

Este cuadro resume cómo se establece el bloque de parámetros para A=5 (de lectura de bytes de la memoria del procesador E/S) y para A=6 (de escritura).

Si la dirección sólo va a ser de 16 bits, que será el caso para el procesador de entrada/salida en un BBC Micro estándar, entonces la dirección se almacena con su byte inferior en (XY+0) y su byte superior en (XY+1).

- *OSWORD con A=7 y A=8*: Estas dos llamadas permiten al programador manipular el sonido con programas escritos tanto en lenguaje máquina como en BASIC. OSWORD con A=7 simula la orden SOUND del BASIC, y OSWORD con A=8 es el equivalente de una orden ENVELOPE. Veámoslas separadamente.
- *OSWORD con A=7*: Exige un bloque con ocho

bytes (véase su estructura en el cuadro) además de una indicación que le permita distinguir qué entrada del bloque corresponde a cada uno de los parámetros dados en la instrucción SOUND.

Bloque de parámetros para OSWORD con A=7	
Byte	**Función**
0	BmnS de NÚMERO CANAL
1	BmsS de NÚMERO CANAL
2	BmnS de AMPLITUD
3	BmsS de AMPLITUD
4	BmnS de TONO ("PITCH")
5	BmsS de TONO
6	BmnS de DURACIÓN
7	BmsS de DURACIÓN

El programa que ofrecemos a continuación ofrece la actuación de una llamada: simula la instrucción SOUND 1,−15,100,30.

```
10 DIM C 20
20 X%=C MOD 256
30 Y%=C DIV 256
40 A%=7
50 C?0=1
60 C?1=0:REM Canal 1
70 C?2=-15 MOD 256
80 C?3=-15 DIV 256
90 C?4=100
100 C?5=0
110 C?6=30
115 C?7=0
120 CALL &FFF1
130 END
```

Naturalmente, este mismo efecto se consigue con muchísima mayor facilidad por medio de la misma instrucción SOUND, por lo que, en BASIC, esta llamada es de rarísimo uso.

• *OSWORD con A=8*: Esta llamada es todavía más complicada que OSWORD con A=7: ¡nada menos que 17 son los bytes que necesita en el bloque de parámetros!

Como siempre, los registros X e Y sirven para apuntar al bloque. La primera entrada dentro del bloque retiene el número de ENVELOPE, y los trece bytes restantes del bloque retienen los demás parámetros de ENVELOPE. Con toda seguridad usted encontrará mucho más fácil el empleo de la instrucción ENVELOPE desde BASIC para conseguir el sonido que desea.

Byte	Valor
0	2
1	4
2	0
3	0
4	0

Las llamadas OSWORD para gráficos

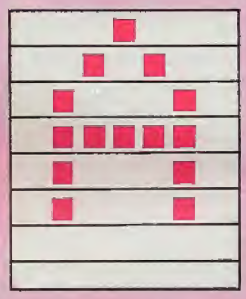
• *OSWORD con A=9*: Funciona de manera equivalente a la instrucción POINT del BASIC pero dentro de un programa en código máquina. Produce el color lógico en un punto dado, o el valor 255 si el punto especificado no está en pantalla. El valor se da en el quinto byte del bloque, que se establece así:

Bloque de parámetros para OSWORD con A=9	
Byte	**Función**
0	BmnS de la abscisa X
1	BmsS de la abscisa X
2	BmnS de la ordenada Y
3	BmsS de la ordenada Y
4	Color lógico

OSWORD con A=10

Si se desea imprimir caracteres grandes en pantalla, esta llamada es de mucha utilidad. Permite obtener la definición actual de cualquier carácter en Modos de 0 a 6. Es necesario un bloque de nueve bytes, como siempre apuntado por los registros X e Y. El primer byte del bloque deberá contener el código ASCII del carácter

Bloque de parámetros para A=10	
Byte	**Función**
0	Carácter escogido
1	Primera línea de defin.
2	Segunda línea
3	Tercera línea
4	Cuarta línea
5	Quinta línea
6	Sexta línea
7	Séptima línea
8	Última línea de defin.



Los valores producidos en el bloque de parámetros representan una expresión binaria de la definición del carácter donde 1 es un punto luminoso (pixel) sobre la pantalla y 0 es un pixel no iluminado

• *OSWORD con A=11*: Esta llamada nos permite inspeccionar los colores especificados en el conjunto de instrucciones de la VDU 19; nos permite encontrar el color *físico* asociado a un determinado color lógico. Lo cual puede resultar útil si se olvidaron los colores especificados en una determinada llamada a la VDU 19. Necesita un bloque de 5 bytes, y al realizarse la llamada el primer byte del bloque retendrá el valor del color lógico que nos interesa. Un ejemplo de su uso puede ser si suponemos ejecutada la orden VDU 19, 2, 4, 0, 0, 0. OSWORD con A=11 y un valor de 2 en el primer byte del bloque producirán los valores que mostramos al margen de esta página.

• *OSWORD con A=12*: Es una versión más rápida de VDU 19.

• *OSWORD con A=13*: Es otra llamada con aparentemente escasos usos. Produce las coordenadas X e Y de los últimos dos puntos repasados por el cursor de gráficos, lo que puede resultar práctico en ciertas ocasiones.

De este breve resumen usted puede colegir que las llamadas OSWORD son muy versátiles y proporcionan un fácil manejo de muchas facilidades del BBC Micro.

Tablones de anuncios

Finalmente nos ocuparemos de las numerosas actividades que se desarrollan a través de los tablones de anuncios

Steve Cross

El tablón de anuncios tuvo su origen en un club de ordenadores de Estados Unidos. ¡El club decidió que el tablón de corcho que estaba utilizando para colgar los anuncios del club no estaba a la altura de su imagen de alta tecnología! Algunos socios decidieron que no sería difícil diseñar un sistema de anuncios mediante llamada telefónica, a través del cual el club pudiera visualizar la información para sus miembros. Otras facilidades que se pensó que serían útiles fueron que los socios pudieran intercambiar mensajes, tanto públicos como privados, y cargar y descargar programas de la biblioteca de software del club. Había nacido el tablón de anuncios.

En la actualidad existen centenares de tablones de anuncios a lo largo y a lo ancho del mundo. Algunos los dirigen clubs y grupos de usuarios, pero la gra mayoría los llevan aficionados particulares. Casi todos estos tablones de anuncios están abiertos a cualquiera que desee utilizarlos, y el acceso es gratuito. Uno simplemente debe marcar el número, entrar su nombre y responder a unas pocas preguntas relativas a su terminal.

Existen varios sistemas de tablones de anuncios. Los dos más comunes son The Bread Board System (TBBS) y el Central Bulletin Board System (CBBS). Los sistemas CBBS se intercambian sus listas de usuarios, de modo que, si usted se registra en uno, automáticamente se registrará en todos los otros tablones CBBS. Numerosos tablones funcionan basados en software de diseño casero. En Gran Bretaña el sistema que goza de mayor popularidad es el tablón TBBS. En la actualidad hay 10 de ellos en funcionamiento. Los sistemas TBBS están divididos en cuatro áreas principales: mensajería general, grupos de interés especial (SIG), correo electrónico y transferencia de archivos.

La facilidad de mensajería general es para men-

¿Algún mensaje?
Los tablones de anuncios por ordenador les ofrecen a los usuarios diversos servicios, incluyendo mensajes privados o de carácter general, juegos e información técnica y sobre productos. Su principal objetivo es constituir un foro a través del cual los usuarios se comuniquen con otros que compartan algún interés común

sajes públicos que no estén relacionados ni con un tema ni con una máquina específicos. Los mensajes dejados en esta área incluyen solicitudes de ayuda técnica (y, si todo va bien, respuestas de otros usuarios), bromas, comentarios sobre máquinas nuevas y modems, trucos y pistas, noticias del club, informes sobre libros y revistas y cotilleo en general.

Los grupos de interés especial ofrecen mensajes públicos sobre máquinas específicas y áreas de interés bien definidas. Los SIG típicos son BBC, Tandy, Commodore, CP/M, Modems, ofertas y demandas. Los SIG ofrecidos varían de un tablón a otro, y si usted quisiera que se estableciera un SIG nuevo, podría dejar un mensaje para el *sysop* (*system operator:* operador del sistema) pidiéndole que abriera uno. Los SIG operan a tenor de la demanda: si suscitan interés suficiente, el *sysop* los mantiene.

El correo electrónico, al cual se suele aludir simplemente como E-mail (correo E), permite la transmisión de mensajes confidenciales entre dos usuarios. Los mensajes enviados a través del correo E sólo los pueden leer el autor, el destinatario y el *sysop*, manteniéndose la reserva mediante contraseñas. Cuando uno se registra, por lo general se le pide que elija una contraseña. Ésta se almacena junto con su nombre y le será solicitada cada vez que se conecte con el sistema.

El área de transferencia de archivos permite que los usuarios intercambien software de dominio público. Está diseñado para personas que han escrito un programa en BASIC y desean compartirlo con otros usuarios. Usted no cobrará ningún dinero por ceder sus programas, ni tampoco se le cobrará ninguna cantidad por acceder a las contribuciones efectuadas por otras personas. Al igual que los SIG, esta área normalmente está dividida por tipos de máquina. Por consiguiente, se le preguntará qué ordenador posee y se le ofrecerá luego un menú de programas para esa máquina.

Existen otras facilidades que también varían de un tablón a otro; p. ej., TBBS London ejecuta un juego Diplomacy electrónico, y TBBS Nottingham posee un juego de aventuras en tiempo real. La mayoría de los tablones disponen de un mensaje de presentación para los usuarios nuevos, que se les ofrece automáticamente a éstos cuando se conectan por primera vez; la información adicional suele incluirse en archivos <I> (información).

Datos para comunicaciones de "Mi Computer"	
Tablones de anuncios	
Número de teléfono (datos)	Ver lista
Número de teléfono (preguntas)	No asignado
Costo	Gratuito
Horario de servicio	Variable
Velocidad (baudios)	300/300 (también 1200/75)
Paridad	Ninguna
Bits de datos	8
Bits de final	1
Para obtener ayuda	Generalmente <I>nfo
Para desconectar	Generalmente <G>oodbye
¿Correo electrónico?	Sí
¿Mensajería pública?	Sí
¿Software comercial?	No
¿Software gratuito?	Sí

Tablones en funciones

Los tablones de anuncios suelen aparecer y desaparecer, de modo que no tendría sentido presentar una lista completa. Los ejemplos británicos que incluimos es probable que funcionen durante algún tiempo

Tablón	Horas	Teléfono
TBBS London	24	01-348 9400
Mailbox-80 Liverpool	24	051-428 8924
The Blandford Board	24	0258-54494
CBBS Surrey	24	04826-25174

Conectando

Una vez que haya conectado con un tablón, encontrará que la mayoría de las cosas son autoexplicativas; he aquí algunas útiles pistas:

- Compruebe con atención el horario de servicio. A los principales tablones se puede acceder durante las 24 horas, pero otros sólo están disponibles durante un período limitado. En caso de duda, llame durante las horas del día. Algunos tablones operan con "doble llamada": deje que el teléfono suene una vez, cuelgue y vuelva a llamar.
- Asegúrese de que su terminal esté preparado antes de llamar. La mayoría de los sistemas cuelgan automáticamente si no reciben un tono de portadora dentro de los cinco segundos después de haber respondido. Algunos tablones tardan más en contestar que otros, y si la llamada anterior acaba de terminar, el sistema puede que tarde hasta tres minutos antes de reciclarse y quedar listo para la siguiente llamada.
- Esté preparado para escuchar una voz humana; hasta los tablones de 24 horas necesitan un mantenimiento diario. Al *sysop* le resultará muy molesto responder y escuchar que quien llamaba ha colgado porque esperaba un tono de portadora. Limítese a preguntar si el tablón está disponible.
- Si el número da ocupado, siga probando. La mayoría de los usuarios se limitan de 15 a 30 minutos por llamada, y los tablones con mayor afluencia poseen sólo unos pocos minutos entre llamada y llamada.
- Cuando establezca conexión, no emplee un nombre falso sin avisar al *sysop*. No deje mensajes que puedan resultar ofensivos a algunos usuarios; si lo hace, quizá la próxima vez que llame se encuentre con que el *sysop* ha borrado su nombre de la lista. Para desanimar a los bromistas, muchos sistemas no le permitirán utilizar por completo el tablón durante su primera llamada (sólo se le permitirá, p. ej., acceder a la lectura). Normalmente al día siguiente tendrá un mejor acceso si utiliza la facilidad <N> (nuevo usuario) o <R> (registro).

Sesión de muestra en un tablón de anuncios

Bienvenido al Blandford Board

Operativos los días las 24 horas

CCITT a 300 baudios
8 Bits - sin paridad - 1 de final

Miembro de A.F.P.A.S.

Nombre? MI Computer
Apellido? Ed. Delta
Llamando desde (ciudad)? Barcelona

TBBS saluda a MI COMPUTER
ED. DELTA
que llama desde Barcelona
Correcto s

Anchura del terminal (10-132)? 52

<A> TRS-100 <B> TRS-1/3/4 <C> APPLE
<D> ATARI <E> IBM-PC <F> OSBORNE
<G> BBC <H> APRICOT <I> COMMODORE 64

Entre la letra de su terminal, o <CR> : <CR>

Su terminal, imprime minusculas? s
Su terminal, necesita saltos de linea (LF)? s
Cuantos nulls (1–50)? 0

The Blandford Board

<N> reg. de nuevo usuario
<B> anuncios
<M> seccion de mensajeria
<U> area de utilidades
<I> informacion
<C> charlar con sysop
<G> goodbye para terminar

Instruccion: m

Seccion de mensajeria

<O> Abrir area de mensajes
<M> Correo-privado y confidencial
<S> SIGs de interés especial
<B> Ordens. Blandford (Os House)
<U> Area de utilidades
<E> Salida de seccion mensajeria

Instruccion: o

Abrir area de mensajes

<O> Exam. rapida de cabeceras de mensajes
<R> Leer mensajes
<L> Dejar un mensaje

Al obtener un tono de portadora, pulse <CR> una o dos veces: obtendrá el mensaje para identificación:

Seguir los menús le resultará fácil. Si no está seguro de lo que significa algo, ¡pruebe y se enterará!

Hay mensajes breves y largos. Pulsando "s" normalmente se puede saltar el texto no deseado. Luego se le pide que se conecte entrando su nombre y apellido, y la ciudad desde donde llama

El tablón hace que ratifique esta información, puesto que podría haber dos usuarios con el mismo nombre
OK

Se le solicita información sobre su terminal. Aparte de la anchura de la pantalla, las respuestas dadas aquí se adecuarán a la mayoría de los micros:

Si su terminal no está incluido, pulse <CR>:

Se le pedirá que elija una contraseña para las futuras conexiones (para evitar que otras personas lean mensajes dirigidos a usted) y quizá se le ofrezca información acerca del sistema. Se le informa si hay algún mensaje y se le ofrece el menú de apertura

Y así sucesivamente. Se le enseñan todos los mensajes encabezados por "Commodore". Para salir, pulse "s" de "stop". Esto funciona en muchas partes del tablero, junto con "p" de "pause" y <CR> para recomenzar

En la mayoría de los tablones tendrá que registrarse antes de que se le permita utilizar todo el sistema. La primera vez es muy probable tener acceso sólo a la lectura. Una vez hecho esto, pase al menú de mensajería:

Ésta es la lista típica de grupos de interés especial (SIG):

<M> Correo-privado y confidencial
<E> Salir del area de mensajes
<G> Goodbye para terminar

Instrucción: r

Pulse P (detenerse), S (parar), N (saltar al sig. msj)

<F> Adelante o R atras multiple
<N> Nuevos mensajes
<M> Mensajes marcados
<S> Recuperacion selectiva
<I> Mensaje(s) individual(es)
<A> Abortar recuperacion

Cual? s

Seleccione <F> de; <To> para o <S> tema s
Entrar texto a seleccionar por la palabra:
Commodore

Responder a opcion msj (S/N)? s

Msj£: 3330 *OPEN*
07/11/84 20:07:09 (Leido 14 veces)
De: NORM PHILLIPS
Para: CUALQUIER USUARIO DE COMMODORE 64
Tema: COMMODORE GEN: COMO NUEVO USUARIO DE ESTE SISTEMA AGRADECERIA LA AYUDA QUE ALGUIEN PUEDA OFRECERME PARA APROVECHAR EL SISTEMA AL MAXIMO CON MI COMMODORE. TAMBIEN ME INTERESARIA SABER SI HAY ALGUN OTRO USUARIO EN MI AREA LOCAL. Y SI ASI FUERA, ¿PODRIA PONERSE EN CONTACTO CONMIGO?

<N> Sig msj, <R> Responder, o <S> stop? R

Usted no esta autorizado para escribir mensajes en este tablon

El msj tiene respuestas, las lee ahora (S/N) n

Al principio de cada llamada usted está en condiciones de acceder a todos los tablones combinados. Puede activar el estado de activación de un tablón pulsando la tecla que se indica en el menú de abajo. Los tablones que no se incluyan como activos no le visualizarán mensajes

Tablones combinados activos: Ofertas y demandas, TRS-80, BBC, APPLE, MODEMS, ATARI, C'64
<A> Ofertas y demandas <B> TRS-80
<C> BBC <D> APPLE
<E> MODEMS <F> ATARI
<G> C'64 <CR> para salir

Cual?

Ian McKinnell

El camino crítico

Gracias a "MacProject", hasta quienes carezcan de aptitudes gráficas podrán preparar un plan de proyecto pulcro y asequible

El análisis del camino crítico (CPA: *critical path analysis*) es una poderosa herramienta de dirección de proyectos, que cumple con la definición del *Longman dictionary of business English* (1982), según la cual un proyecto "se divide en diversas etapas separadas, cada una de las cuales se representa en una gráfica mediante una flecha que posee un valor de tiempo. El resultado es una serie de cadenas paralelas que se unen en ciertos puntos. La cadena que lleva el mayor tiempo es el *camino crítico*, porque determina la duración total del trabajo. La gráfica muestra los puntos en los que se necesitarán diversos materiales, herramientas, etc., cuándo debe iniciarse y terminarse cada etapa y por quién".

Se han efectuado intentos por producir programas CPA para ordenadores, pero, puesto que una red CPA es básicamente un concepto gráfico, las dificultades propias de dibujar y redibujar una red cada vez que se entra nueva información han disuadido al programador, y los programas tienden a ser basados en texto y más bien complejos. Esto es contraproducente: el objetivo primordial de una red CPA es representar una secuencia de acciones compleja de una forma que sea inmediatamente accesible. Hasta ahora los programa CPA basados en ordenador han tendido a complicar el resultado en vez de simplificarlo.

El Apple Macintosh, basado en gráficos, es particularmente idóneo para esta clase de trabajo, y *MacProject* hace que la creación de una red (o "gráfica de proyecto", como la denomina el programa) resulte tan sencilla que efectivamente simplifica la dirección de un proyecto. Las fechas y las horas se pueden cambiar a voluntad y ello comporta un nuevo cálculo instantáneo de los tiempos totales del proyecto; se pueden insertar nuevas tareas y suprimir las redundantes, y se pueden asignar, controlar y reasignar recursos, todo ello en un tiempo no superior al que lleva ubicar el ratón del Macintosh y esperar que el *Mac* traiga la rutina necesaria del disco.

El manual, claro y conciso, alude al sencillo ejemplo de una barbacoa familiar para ilustrar el concepto básico, dividiendo el proyecto en cinco tareas:

PREPARAR UNA ENSALADA
TENDER LA MESA
ENCENDER EL FUEGO
COCER EL ARROZ
ASAR LA CARNE

que conducen al fin del proyecto:

CENAR

La mayor parte de las tareas se pueden efectuar de forma concurrente, pero es imposible asar (al gril) la carne hasta que no se haya encendido el fuego, de modo que el camino crítico va desde "encender el fuego", pasando por "asar la carne" hasta "cenar". Si se tarda en encender el fuego, entonces se demorará la cena, aun cuando se hayan completado a tiempo las otras tareas. Del mismo modo, si la persona encargada de preparar la ensalada llega tarde, la elaboración de la ensalada podría volverse crítica y los hambrientos comensales tendrían que esperar a la condimentación de la ensalada (y la carne se enfriaría, se quemaría o incluso se arruinaría).

Los mismos principios que han sido válidos para preparar una barbacoa se pueden aplicar a cualquier proyecto, desde organizar un congreso empresarial hasta construir una refinería de petróleo.

Los casilleros de las tareas de la gráfica se dibujan desplazando el patrón. Cuando se inicia la preparación de la gráfica, éstos se fechan a partir del 2 de enero, pero se pueden cambiar las fechas, ya sea eligiendo una nueva o bien, si existe un límite de tiempo, eligiendo una fecha de finalización. Una vez creado el primer casillero, se pueden unir al mismo otros casilleros del mismo tamaño desplazando al ratón desde el casillero anterior hasta la posición deseada. Pulsando el botón del ratón dentro de un casillero se pueden entrar las etiquetas de las tareas; el programa centra el texto de forma automática dentro del espacio disponible. Los casilleros se pueden ensanchar pulsando el botón del ratón en el margen, visualizándose entonces ocho tamaños. Desplazando el cursor hasta uno de ellos e inclinando el ratón, se modificará el tamaño. Desplazando el cursor hasta el margen pero no hasta un tamaño, y moviendo el ratón, el casillero se puede trasladar por la gráfica.

Para asegurar una gráfica elegante y bien cuadrada, se debe seleccionar INVISIBLE GRID (cuadrícula invisible) de la opción del menú LAYOUT antes de dibujar el casillero de la primera tarea. Entonces todos los casilleros se alinearán con exactitud.

Ian McKinnell

Opciones de pantalla
El calendario *MacProject*, que vemos aquí como la ventana superior de la pantalla de un Macintosh, permite entrar la cantidad de horas y de días de trabajo en los cuales se pretende completar el proyecto, además de tener en cuenta los días de vacaciones, los fines de semana y otras consideraciones especiales. Debajo del mismo está la hoja Cash Flow, que lleva la cuenta actualizada de los gastos del proyecto

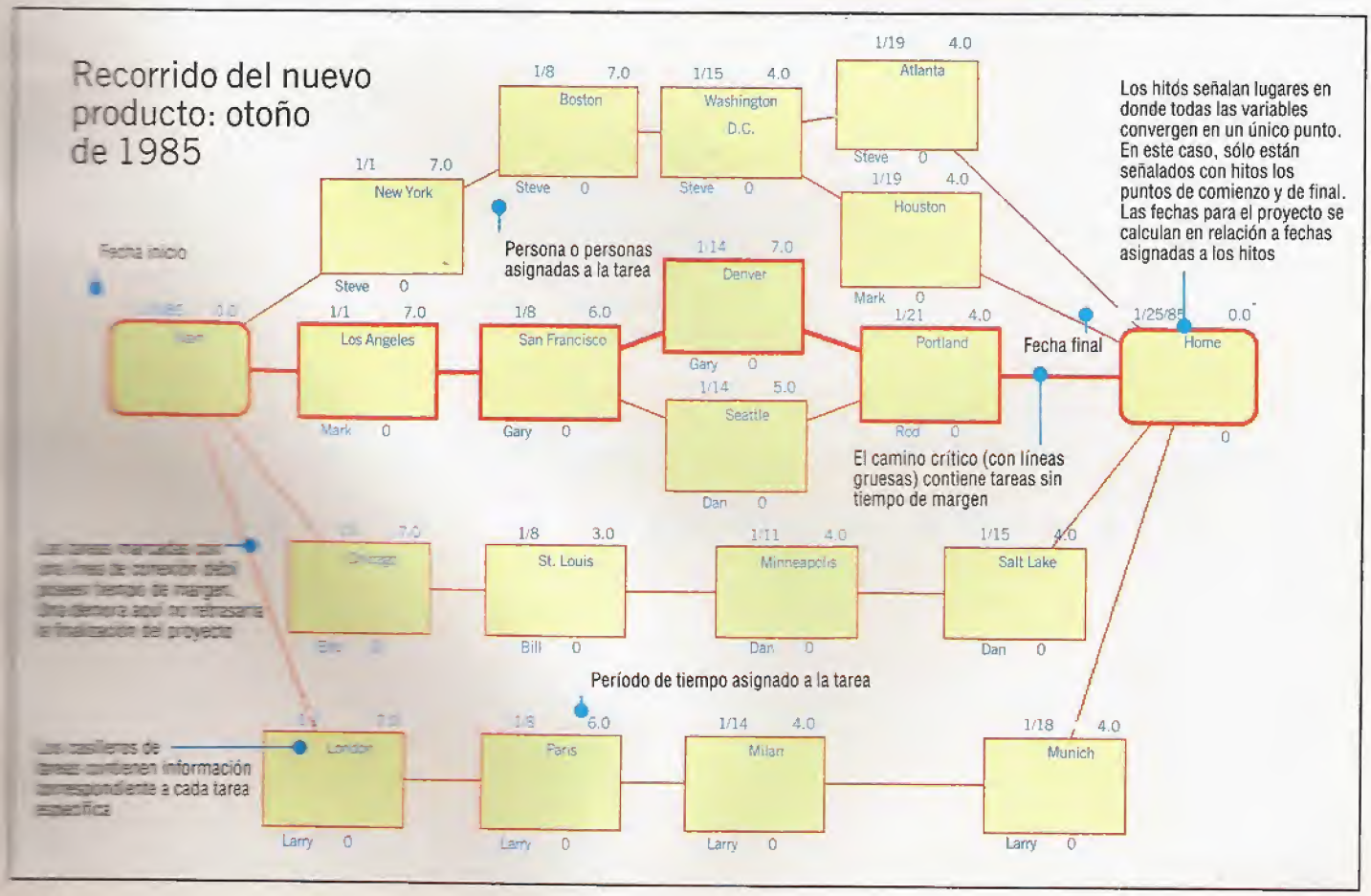
Cuando se ha etiquetado un casillero, se puede entrar TASK INFO desde el elemento apropiado del menú. Ésta permite entrar la hora (en días, minutos, semanas o lo que fuere) y recursos, que pueden ser personas o suministros. Asimismo, en la gráfica principal se pueden visualizar hasta cuatro elementos de información alrededor de cada casillero de tareas: Resource (recurso), o Earliest Start (inicio más temprano) arriba a la izquierda; Duration, Resource o Earliest Finish (duración, recurso o terminación más temprana) arriba a la derecha; Latest Start, Fixed Cost o Resource (inicio más tardío, costo fijo o recurso) abajo a la izquierda; y Latest Finish, Fixed Income o Resource (terminación más tardía, ingreso fijo o recurso) abajo a la derecha. Una vez completada la gráfica del procedimiento, el camino crítico se indica con una línea más gruesa y los caminos no críticos mediante líneas más débiles.

Mientras tanto, sin que el usuario haya emprendido ninguna otra acción, se han preparado una serie de gráficas de análisis. RESOURCE TIMELINE visualiza una gráfica de barras que muestra cómo se han asignado los recursos. TASK TIMELINE es similar, pero lista las tareas en vez de los recursos (o personal) asignados a cada una. En color gris se indica el "tiempo de margen" (la diferencia entre el tiempo de finalización más temprano y el más tardío, durante el cual se puede completar la tarea sin demorar el proyecto total). PROJECT TABLE produce una tabla de todas las tareas, mostrando las tareas críticas en negrita y visualizando la cantidad de días asignados, inicio y finalización más tempranos, inicio y terminación más tardíos, costos fijos, costo de recursos, ingresos fijos y nombres de recursos para cada uno. Hay tres tablas que relacionan los costos, explicándose por sí solos los fines de cada una: TASK COST ENTRY (entrada de costos de tareas), RESOURCE COST ENTRY (entrada de costos de recursos) y CASH FLOW TABLE (tabla de movimiento de caja).

Desde la opción LAYOUT (trazado) del menú se puede reflejar un diagrama esquemático de toda la gráfica (sin texto explicativo) o modificar las dimensiones de ésta. Si se ha suprimido una cantidad de casilleros al final de un diagrama, el espacio vacío se seguirá viendo y esta opción permite condensar la gráfica adecuadamente.

MacProject es una eficaz implementación de los principios del análisis del camino crítico, que se le explican al lector profano en un lenguaje claro y fácil de comprender, sin emplear la jerga (el CPA ni siquiera se menciona). Las capacidades para gráficos del Macintosh se utilizan al máximo y de modo tal que resulta difícil imaginar que pueda salir al mercado una implementación similar para cualquier otra máquina. Es poco probable, por supuesto, que un usuario se sienta tan atraído por la sofisticación del software como para adquirir el ordenador para ese solo uso.

MacProject: Para el Apple Macintosh
Distribuido por: Apple Computer (UK), Eastman Way, Hemel Hempstead, Herts HP2 7HQ, Gran Bretaña
Autores: Debra Willett y Stephen D. Young, de Solosoft
Formato: Disco



Ejemplo práctico
Esta gráfica del *MacProject* muestra un recorrido por diversas ciudades en las que una determinada empresa se encuentra promocionando un nuevo producto. Al proporcionar una relación visual del viaje, *MacProject* hace más expedita y sencilla la asignación de personal y tiempo al proyecto

Un profesor severo

Comenzamos una serie dedicada al PASCAL, el más prestigioso de los lenguajes de programación de alto nivel

Principales características de programación del PASCAL
- Formato y trazado del texto completamente libres
- Flexibilidad en cuanto a los nombres de los objetos
- Capacidad para definir nuevas palabras instrucción
- Sintaxis simple y coherente
- Digitación de datos
- Estructura de programación modular
- Control flexible de datos y procesos
- Recursión natural
- Excelente diag. de errores en tiempo de compilación
- Compilador pequeño y sumamente eficaz

El lenguaje de programación PASCAL lo inventó el profesor Niklaus Wirth, de Zurich, aproximadamente en 1970. Debe su nombre al matemático francés del siglo XVII Blaise Pascal, quien inventara la primera calculadora de cuatro operaciones. El lenguaje PASCAL se vio influido fundamentalmente por el ALGOL 60 y fue una réplica directa de Wirth al complejo y enorme ALGOL 68. Wirth pretendió que el PASCAL:

- Permitiera la expresión exacta de conceptos y estructuras de programación.
- Demostrara que un lenguaje pequeño e independiente de la máquina, con un conjunto de datos, sentencias y estructuración de programa, se podía utilizar como una herramienta para resolver problemas de carácter general.
- Contribuyera a profundizar en los métodos de organizar programas extensos y dirigir proyectos de software complejos con firmeza y seguridad.
- Tuviera amplias capacidades para verificación de errores, especialmente durante la compilación, minimizando, por consiguiente, los errores de programación y proporcionando un vehículo excelente para enseñar programación de ordenadores.
- Se pudiera implementar eficazmente en microordenadores.

Todos los objetivos de diseño se han cubierto con gran éxito: un pequeño compilador de PASCAL ocupa típicamente 24 K y es dos veces más eficaz que el FORTRAN (notable por su velocidad). Aunque el PASCAL posee un vocabulario pequeño y es fácil de aprender (posee sólo 35 palabras "clave" o "reservadas", frente al más de un centenar que poseen la mayoría de las variantes del BASIC), es, no obstante, mucho más eficaz y, más importante aún, mucho más expresivo tanto en la forma en que se pueden escribir los algoritmos como en la facilidad con que se pueden describir los datos de forma simple y coherente, independientemente de lo complejos que puedan ser.

Tal vez la mayor ventaja del PASCAL sea el hecho de que proporciona una forma simple y coherente de expresar algoritmos sumamente potentes. Más que ninguna otra cosa, lo que influye más directamente en la sencillez con la cual podemos resolver los problemas informáticos es la *forma* en que uno piensa en los mismos. La natural libertad de expresión del PASCAL significa que el lenguaje es una maravillosa herramienta para resolver problemas, en vez de ser parte del propio problema. El lenguaje posee, asimismo, muchas otras ventajas. El PASCAL es un lenguaje compilado, lo que significa no sólo que los programas se ejecutan muchas veces más rápido, sino que el texto del programa fuente y el intérprete del lenguaje no ocupan una preciosa memoria: todo lo que se necesita es el código "objeto" compilado.

La filosofía global del lenguaje es proteger al programador de su propia torpeza e impedir absolutamente que funcione un programa plagado de errores. Esto puede parecer muy extraño a primera vista, particularmente para los programadores de BASIC. En BASIC, sin embargo, es frecuente que cuanto más rápido conseguimos ejecutar un programa, más tiempo pasa hasta que conseguimos hacerlo funcionar correctamente.

De hecho, en especial para un programa largo, programar en PASCAL es mucho más fácil que programar en BASIC. En algunas ocasiones se dice que el PASCAL es un "profesor severo" y, aunque a veces esto se plantea como una crítica, de hecho evidencia la excelencia de la capacidad de diagnóstico e informe de errores del PASCAL. A veces necesitamos una bofetada para recordar que escribir programas robustos libres de errores exige concentración y cuidado. Al fin y al cabo, estas disciplinas impuestas no son más que las que se requieren para organizar con eficacia y seguridad cualquier programa. La resultante reducción del tiempo invertido para depurar programas que "ya casi funcionan" es una verdadera ventaja. En resumen, el PASCAL ayuda a hallar soluciones: no forma parte del problema en sí mismo.

Para el programador que llega al PASCAL desde el BASIC, la diferencia más llamativa de los programas en PASCAL, de cualquier tamaño, es la abundancia de definiciones y declaraciones de aspecto peculiar que parecen tener poco sentido, cuando no ser completamente inútiles. El primer palmo y medio del fuente de cualquier programa largo en PASCAL parece realmente no hacer nada. Ello se debe en parte a que, si bien uno efectivamente puede añadir al lenguaje sus propias palabras, hay que hacerlo antes de utilizarlas, para que el PASCAL pueda comprender lo que significan. Por consiguiente, mientras que en BASIC primero se enuncia un programa (empleando sentencias como GOSUB 5000) y las subrutinas se definen *después* del programa principal, el PASCAL permite que usted defina instrucciones nuevas tales como Limpiar-Pantalla o Pausa (de tantos segundos) al principio del programa y utilizarlas después en el procedimiento principal. Por ejemplo:

```
begin
   LimpiarPantalla;
   Escribir ("Hola!");
   Pausa (3);
   ...etc.
```

El PASCAL propicia un enfoque metódico, pero muy práctico, a la programación de ordenadores. Presenta al usuario una visión conceptualizada, de muy alto nivel, de lo que es un sistema informático, de modo que los datos y los procesos se pueden definir y expresar de una forma lógica y natural. Ello ase-

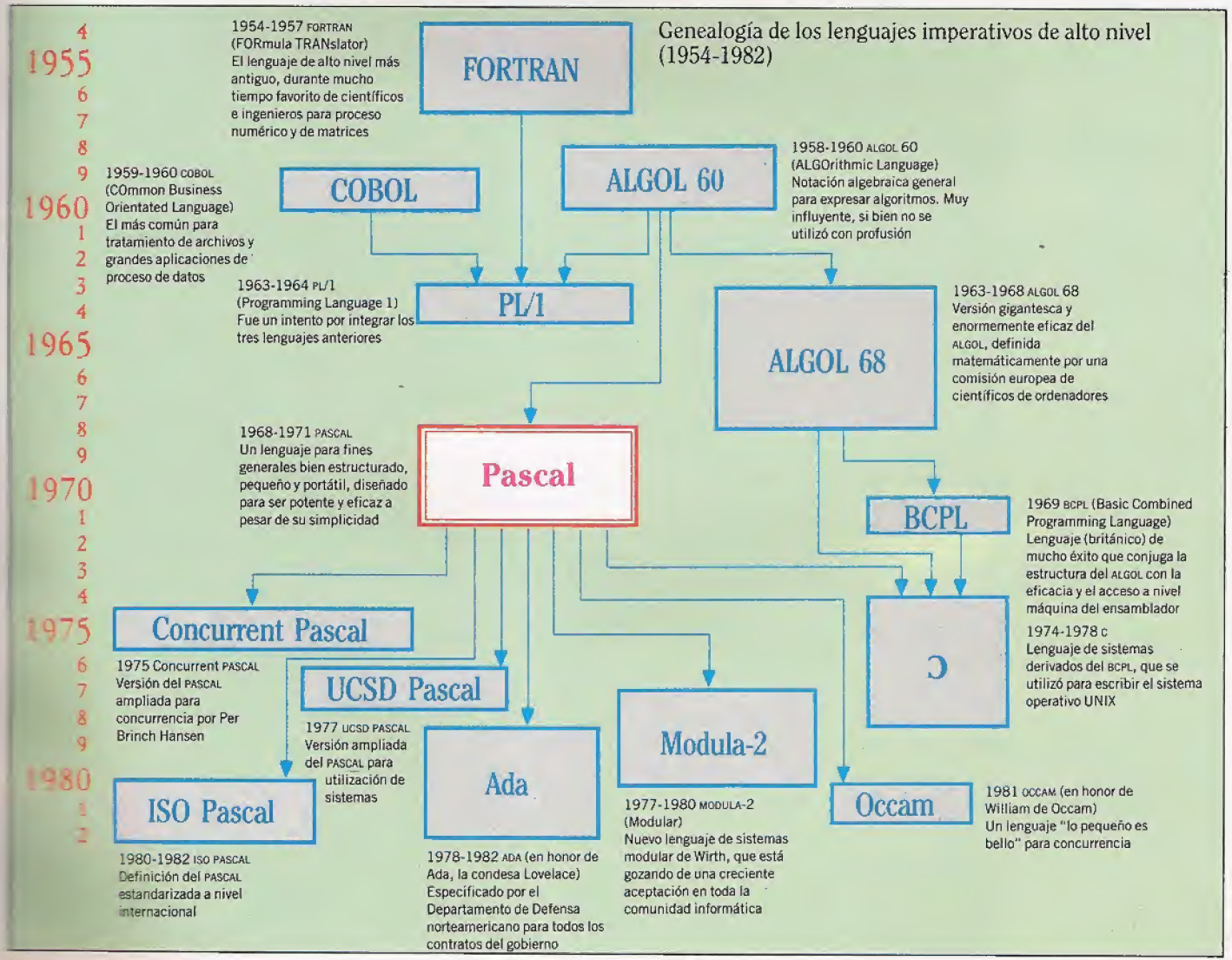
Blaise Pascal (1623-1662) Matemático, científico y filósofo, Blaise Pascal diseñó en 1642 la primera calculadora mecánica del mundo. El lenguaje PASCAL recibió su nombre como homenaje a su contribución a la ciencia informática

gran un elevado nivel de portabilidad y seguridad implícitas, y tanto la detección como el diagnóstico de errores son excelentes. En general, el PASCAL ha ejercido sobre los otros lenguajes de ordenador (y sobre el diseño del software) una influencia mayor que cualquier otro lenguaje de programación.

Nuestro diagrama, que ilustra la genealogía de los lenguajes de alto nivel, refleja solamente las influencias más importantes de la mayoría de los lenguajes imperativos compilados importantes, y por ello no incluye el LISP, el PROLOG ni ningún otro lenguaje funcional. Tampoco incluye al FORTH, ¡porque es prácticamente inclasificable! La corriente principal de influencia comienza con el ALGOL 60, y casi no existe ningún lenguaje moderno, de ninguna clase, que no haya derivado, ya sea directa o indirectamente, del PASCAL. Esto a su vez significa que contar con un conocimiento profundo del PASCAL será una enorme ventaja para entender los lenguajes de la próxima década, com el MODULA-2, el OCCAM o el ADA.

Además de utilizarse con profusión en la enseñanza, el PASCAL ha sido ampliamente adoptado para muchas aplicaciones comerciales y de sistemas. Se lo ha utilizado para escribir software tan diverso como paquetes financieros y compiladores de lenguajes. El *p-system*, el famoso sistema operativo portátil diseñado en la UCSD (*University of California at San Diego*) a finales de los años setenta, se desarrolló y se escribió en PASCAL. El software para el Lisa y el Macintosh de Apple, incluyendo sus sistemas operativos, se escribió fundamentalmente ya sea en PASCAL, o en su derivado, el CLASCAL. Muchos miles de programadores profesionales se formaron con el PASCAL, ya sea en el DEC o el VAX de la Universidad o, posiblemente, en un Apple II ejecutando la versión del *p-system* denominada PASCAL Apple. En la actualidad se puede encontrar a miembros del equipo UCSD original escribiendo afanosamente compiladores de MODULA-2 u otros sistemas y software de aplicaciones para muchas de las principales empresas de software: todo ello en PASCAL o un derivado.

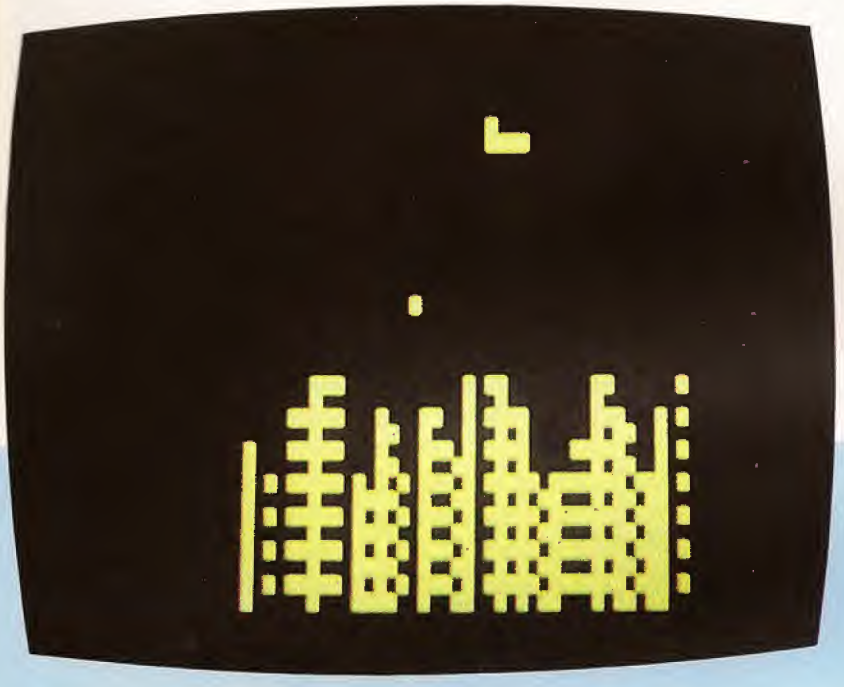
Lo más notable de este fenómeno es el hecho de que ha sucedido a pesar de la más absoluta carencia de patrocinamiento por parte de los representantes de los más importantes intereses comerciales. El PASCAL triunfa por sus propios méritos y no porque ninguno de los grandes fabricantes haya invertido algo en venderlo. Nuestra serie de programación en PASCAL se concentrará en el lenguaje estándar (ISO PASCAL), pero cuando se requieran gráficos o llamadas al sistema operativo se ofrecerán ejemplos específicos para algunas máquinas.



Ian McKinnell

Bombardeo aéreo

Este juego bélico le permite adaptar el nivel de dificultad a voluntad. El listado que ofrecemos es para el micro Alice



Su misión consiste en destruir la ciudad que está sobrevolando, con objeto de poder aterrizar. A cada pasada, su avión vuela un poco más bajo. No puede lanzar una bomba (pulsando una tecla cualquiera) hasta que la bomba precedente haya alcanzado su objetivo o llegado al suelo. Cuando el avión haya aterrizado (o se haya estrellado contra un edificio), queda registrada la puntuación, así como la puntuación más alta del día. Si este juego le parece demasiado difícil, puede variar los límites de la ciudad (6 y 26, línea 60) y la altura máxima de los edificios (reemplazando 8, línea 80, por un valor superior; 10, p. ej.).

```
  5 REM ***************************
 10 REM *     BOMBARDEO AEREO     *
 15 REM ***************************
 16 REM
 17 REM INICIALIZACION
 18 REM
 19 REM A$=AVION
 20 A$=CHR$(128)+CHR$(155)+CHR$(147)
 25 REM B$=BOMBA
 30 B$=CHR$(145)
 35 REM H=POSICION AVION Y MARCADOR
 40 H=0
 50 CLS 0
 55 REM ESTABLECER CIUDAD
 60 FOR I=6 TO 26
 70 C=RND(7)+151
 75 REM REEMPLAZAR 8 POR OTRO
 76 REM VALOR PARA CAMBIAR
 77 REM ALTURA EDIFICIOS
 78 REM LINEA 80
 80 FOR J=15 TO RND(4)+8 STEP-1
 90 PRINT @ J*32+I,CHR$(C);
100 NEXT J
110 NEXT I
114 REM
115 REM BUCLE PRINCIPAL
116 REM
120 PRINT @ H,A$;
125 REM AVION ESTRELLADO?
130 IF PEEK(16387+H) < > 128 THEN
    1000
155 REM TIR
160 IF INKEY$ < > " " AND B=0 THEN
    B=H+33
165 REM BOMBA LLEGA AL SUELO?
170 IF B<> 0 THEN GOSUB 5000
175 REM HA FALLADO?
180 IF B=0 THEN GOSUB 6000
185 REM AVANCE AVION
190 H=H+1
195 REM ATERRIZADO?
200 IF H=507 THEN 1000
210 GOTO 120
994 REM
995 REM ATERRIZADO
996 REM
 999 REM RECORD BATIDO?
1000 IF H>R THEN R=H
1010 PRINT @ 3,"MARCADOR" :"H,
1020 PRINT "RECORD :";R;
1030 FOR I=1 TO 100
1040 NEXT I
1050 R$=INKEY$
1060 PRINT @ 73, "OTRA"?
1070 R$=INKEY$
1080 IF R$=" " THEN 1070
1090 IF R$=<>"N" THEN 20
1100 CLS
1110 END
4994 REM
4995 REM BOMBA LANZADA
4999 REM LLEGA AL SUELO
5000 IF B>=510 THEN B=0:GOTO 5020
5005 REM EDIFICIO ALCANZADO?
5010 IF PEEK(16384+B) <>128 THEN
     PRINT @ B,CHR$(128);
5015 REM ESTABLECER BOMBA
5020 PRINT @ B1,CHR$(128);
5030 IF B<>0 THEN PRINT B,B$;:B
     1=B:B=B+32
5040 RETURN
5994 REM
5995 REM DEMORA
5996 REM
6000 FOR I=1 TO 20
6010 NEXT I
6020 RETURN
```

Largamente esperado

Tras su apariencia de máquina para juegos, el Enterprise Sixty Four oculta poderosas facilidades de programación

Cuando se lanzó el ordenador personal Enterprise Sixty Four, que originalmente iba a llamarse Elan y luego Flan, ya había establecido una especie de récord, aun en una industria acostumbrada a grandes demoras entre el anuncio de un producto y su llegada a los escaparates. El Enterprise tardó casi dos años.

El problema, tal como ocurrió con el QL, fue que el fabricante tuvo problemas para comprimir las especificaciones, realmente notables, en los chips fabricados por encargo. Sin embargo, ahora los problemas ya se han solucionado y la pregunta es: ¿valía la pena esperar tanto?

En su presentación, el Enterprise es, por cierto, inusual, aunque quizá no tan original como prometía ser cuando se anunció la máquina. En vez de la carcasa tipo libro que se ha convertido en la norma para todo ordenador personal, la parte frontal tiene bordes curvados que se extienden hacia la parte cuadrada posterior. El teclado tipo máquina de escribir es del mismo gris humo que el resto de la carcasa.

Las teclas se hallan en todas las posiciones normales. Las de control estándares, como Control, Escape y Enter, también están situadas de forma sensata. El juego de control incluye las poco habituales adiciones de dos teclas Delete (Erase y Delete) que suprimen un carácter hacia atrás y hacia adelante, respectivamente. Hay una tecla Hold, que congela la ejecución de un programa.

Las teclas de control, a los lados de las teclas de máquina de escribir, son de color verde. Encima del teclado hay ocho teclas de función azules, que poseen 16 funciones programadas. Éstas siguen la tendencia actual de programarse con instrucciones utilizadas comúnmente, tales como LIST y AUTO. Sin embargo, son reprogramables.

La tecla Stop roja interrumpe la ejecución de un programa, que se puede reiniciar pulsando RUN o CONT. Esta función por lo general se lleva a cabo mediante una tecla Escape. En el Enterprise la tecla Escape se emplea para llevarlo a uno de un nivel de programa al nivel siguiente.

Las teclas propiamente dichas están ligeramente elevadas respecto a la línea de la carcasa. Aunque este sistema es mejor que el del QL, el moldeado y el tacto de las teclas dan la sensación de ser poco fiables. La experiencia directa, sin embargo, hace desaparecer de inmediato esta impresión, y las teclas resultan ser más fiables que las de algunas máquinas que poseen teclados de aspecto más convencional.

La palanca de mando incorporada mide 3 cm y está montada en un diafragma. A los amantes de los juegos, su reducido tamaño les resultará incómodo para ejecutar los juegos recreativos estilo "matamarcianos".

Sobre el lado izquierdo de la máquina hay una puerta para cartuchos, denominada ROM Bay. A diferencia de las máquinas modernas, el BASIC se suministra en cartucho en vez de estar residente en la máquina; no cabe duda de que Enterprise tiene la intención de ofrecer en un futuro cercano otros lenguajes y aplicaciones en cartucho. Aunque la ranura para cartuchos está un tanto escondida hacia dentro de la máquina, la inserción del cartucho no ofrece mayor problema. En el lado opuesto de la máquina hay un conector marginal en paralelo que aparentemente posibilitará cualquier futura ampliación. Enterprise planea, asimismo, controlar las unidades de disco desde la puerta para ampliación.

Llegada tardía
A pesar de que el ordenador Enterprise Sixty Four se anunció en 1983, la máquina empezó a salir al mercado en grandes cantidades dos años después. La demora, sin duda alguna, le ha costado cara a la empresa, porque, aunque en 1983 las especificaciones de la máquina eran notablemente avanzadas, ahora la competencia ya se ha puesto al día

En la parte posterior hay un botón Reset que, al ser pulsado una sola vez, pone en funcionamiento al ordenador. Cuando se pulsa dos veces indica a éste que cargue un cartucho. Esta última facilidad es necesaria porque, de lo contrario, el ordenador verifica si hay algún cartucho colocado sólo al encenderse.

En la parte trasera de la máquina hay, asimismo, diversas interfaces, incluyendo un par para palancas de mando y una puerta para impresora en paralelo compatible con Centronics. Una puerta en serie/red compatible con RS232/432 permite conectar el Enterprise a una red de área local o a dispositivos en serie, y comunicarse con otros ordenadores a través de un modem externo. Una puerta para pantalla permite que el ordenador soporte una pantalla a color RGB.

Estas interfaces son inusuales por el hecho de estar instaladas como conectores marginales. Sin

embargo, si la máquina obtiene éxito, es seguro que los proveedores independientes ofrecerán interfaces que permitan, por ejemplo, instalar palancas de mando tipo Atari. La idea detrás de todo esto parece permitir la máxima flexibilidad en el uso de periféricos; las puertas tipo Atari, por ejemplo, se limitarían a una función específica, mientras que un conector marginal permite la instalación de dispositivos de control tales como tablillas al tacto. No obstante, cuando la tendencia general en materia de ordenadores es la de "enchufar y usar", el Enterprise parece descolgarse respecto a los otros fabricantes. Por supuesto, esto significa que, al menos de momento, los usuarios se verán obligados a adquirir los propios periféricos de Enterprise en vez de poder elegirlos libremente.

Otros conectores proporcionados en la parte posterior de la máquina son una interface para cassette con dos enchufes REMotos, que le permiten al usuario utilizar dos máquinas de cassette de forma simultánea: una para entrada y otra para salida. Hay también un enchufe RF para conexión a un televisor y el conector de potencia para el transformador externo.

Tras el encendido, la pantalla visualiza el logotipo de Enterprise. El usuario pulsa entonces la tecla Enter y, en el supuesto de que no esté instalado el cartucho de BASIC, se visualiza el procesador de textos incorporado. Éste es un paquete muy útil y es notablemente superior al procesador de textos que lleva incorporado el Commodore Plus/4. Existen numerosas opciones, que le permiten al usuario entrar y formatear el texto. Las funciones utilizadas más comúnmente se visualizan en la parte superior de la pantalla, y a las otras se accede pulsando f5, que ofrece un menú de las otras opciones.

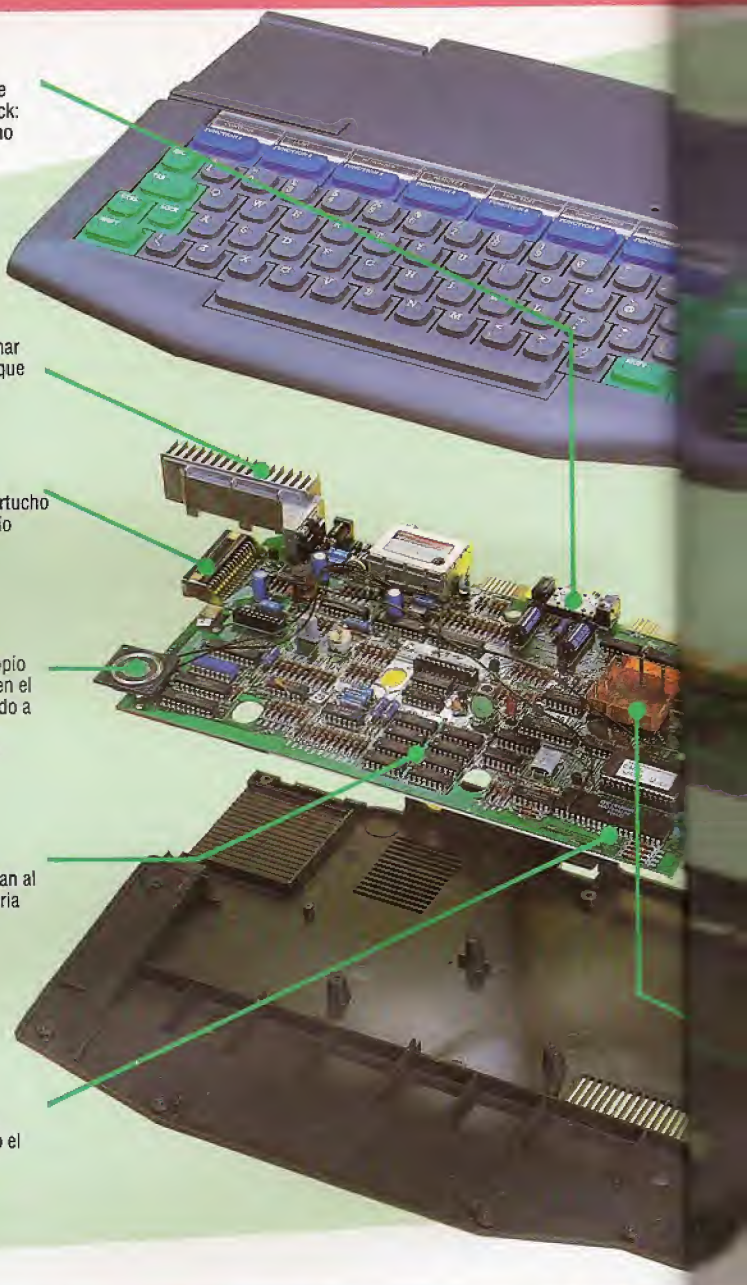
La opción más sorprendente (y mejor recibida) le permite al usuario escribir ya sea en modalidad de 40 columnas (la opción por defecto) o bien en modalidad de 80 columnas. Una opción de 80 columnas, combinada con el procesador Z80, en el cual se basa el Enterprise, indica que una de las primeras provisiones para las unidades de disco, cuando éstas salgan, será el CP/M.

Dado que el Enterprise carece de teclas para el cursor, éste se guía a través de la pantalla mediante la palanca de mando. A muchos usuarios al principio les resultará difícil controlarla, así como posicionar el cursor con exactitud, pero con un poco de práctica se puede llegar a dominar la técnica de editar con la palanca de mando.

El BASIC del Enterprise Sixty Four

Si al encender la máquina está instalado el cartucho de BASIC, cuando se pulsa la tecla Enter la pantalla pasa al BASIC. Aquí se aprecia la ventaja de tenerlo retenido en cartucho en vez de estar residente en el ordenador. Para programar, el usuario tiene a su disposición 49 K completos de memoria. El BASIC, de Intelligent Software, fue una de las primeras cosas que despertaron interés por el Enterprise y por cierto parece ser una de las mejores implementaciones.

El Enterprise posee cuatro generadores de sonido, pudiéndose acceder a cada uno de ellos mediante una instrucción SOURCE. Algunas de las ins-



Interface para cassette
La interface se compone de cuatro conectores microjack: un conector de entrada, uno de salida y dos REMotos

Disipador
Se proporciona para eliminar cualquier exceso de calor que se pudiera producir en el ordenador

Puerta para cartuchos
Aquí se debe instalar el cartucho de BASIC para que el usuario pueda escribir y ejecutar programas en BASIC

Altavoz
El Enterprise posee su propio altavoz incorporado, si bien el equipo puede ser conectado a un sistema *hi-fi* externo a través de la interface para pantalla

Chips de RAM
Estos chips le proporcionan al ordenador 64 K de memoria direccionable

CPU
El Enterprise utiliza como unidad central de proceso el popular microchip Z80A

Software basado en ROM

Pantalla PT 40 columnas

Pantalla PT 80 columnas

Programa para gráficos tortuga

Visual. de gráficos tortuga

A diferencia de la mayoría de los otros ordenadores, el Enterprise posee un procesador de textos incorporado. Si no hay ningún cartucho instalado, el ordenador pasará automáticamente al procesador de textos. Éste se puede utilizar tanto en modalidad de 40 columnas como de 80

Una de las grandes sorpresas del Enterprise es el empleo de gráficos de tortuga. Aparte del empleo de la instrucción PLOT, las instrucciones para gráficos de tortuga se pueden tomar fácilmente por LOGO. Especificando la dirección y la distancia en la cual se ha de dirigir el lápiz se pueden dibujar complicados diseños

Ian McKinnell

Palanca de mando
Para utilizar con el editor de pantalla completo, el Enterprise posee una palanca de mando incorporada en lugar de teclas para el cursor

Mando de Reset
Al pulsarse, la máquina hace un "inicio en caliente"

Puerta para ampliación
Este bus en paralelo se utilizará para la futura ampliación. Aquí se conectarán las unidades de disco

Chip de audio
Este microchip, el segundo de los chips de Enterprise fabricados a medida, se llama *Dave* (este nombre está impreso en el chip). Controla los 4 canales de sonido y las 8 octavas de que dispone el ordenador

Chip de video
Apodado *Nick* en honor del diseñador del chip, este dispositivo fabricado bajo pedido le proporciona al Enterprise sus excelentes gráficos

trucciones disponibles son PITCH, DURATION y ENVELOPE, que permiten al programador modificar la forma de una nota. Dado que el Enterprise puede producir sonido estéreo, las instrucciones LEFT y RIGHT especifican el altavoz desde el cual se ha de emitir el sonido.

Si bien el sonido que producen los chips del Enterprise es excelente, la calidad del altavoz incorporado constituye una de las mayores decepciones respecto a la máquina. Éste es apenas algo mejor que el del Sinclair Spectrum. Sin embargo, se puede conectar el ordenador a un sistema *hi-fi* a través del conector marginal para pantalla.

Las capacidades para gráficos del Enterprise son excelentes. Hay ocho modalides disponibles para gráficos. El usuario puede optar por tener en la pantalla 2, 4, 16 o 256 colores diferentes al mismo tiempo ya sea en modalidad HIRES o LORES. LORES posee sólo la mitad de resolución horizontal que HIRES. La cantidad de colores que se pueden visualizar en cualquier momento varía también según la resolución. Así, HIRES 2 puede producir 2 colores con una resolución de 640 por 180; HIRES 256, por el contrario, puede visualizar los 256 colores pero posee una resolución de sólo 80 por 180 pixels.

Los colores se pueden elegir ya sea seleccionando un número comprendido entre 0 y 255, o bien, en el caso de ocho colores preprogramados, por su nombre; por ejemplo, WHITE o RED. Aún más interesante es el hecho de que la instrucción RGB permite seleccionar colores mezclándolos entre sí. El rojo, por ejemplo, corresponde a RGB(1,0,0) y el verde es RGB (0,1,0). Unidos, ambos producen RGB (1,1,0): amarillo.

Si el Enterprise hubiera salido al mercado en 1983, cuando fue anunciado, habría obtenido un éxito instantáneo; sólo la calidad del software incorporado así lo habría asegurado. Sin embargo, hoy en día la competencia es muchísimo mayor y el Enterprise, a pesar de sus excelentes características, no es hoy más que una de las numerosas alternativas, entre las que se incluyen las nuevas máquinas de Sinclair y Commodore. El tiempo dirá si esta demora ha sido fatal para el éxito del Enterprise.

ENTERPRISE SIXTY FOUR

DIMENSIONES
60 x 260 x 405 mm

CPU
Z80A operando a 4 MHz

MEMORIA
64 K de RAM, de los cuales hay 49 K disponibles para programas en BASIC y 48 K de ROM (32 K para el OS y procesador de textos, y otros 16 K más disponibles a través del cartucho de BASIC)

PANTALLA
Para el procesador de textos, pantalla de textos de 80 x 24 como máximo. Ocho modalidades para gráficos disponibles, dando la más alta una resolución de 672 x 512 pixels. Máximo de 256 colores en la pantalla al mismo tiempo

INTERFACES
Interface en paralelo compatible con Centronics, dos interfaces para palanca de mando, interface en serie compatible con RS232, interface para pantalla RGB, puerta RF y un bus de ampliación

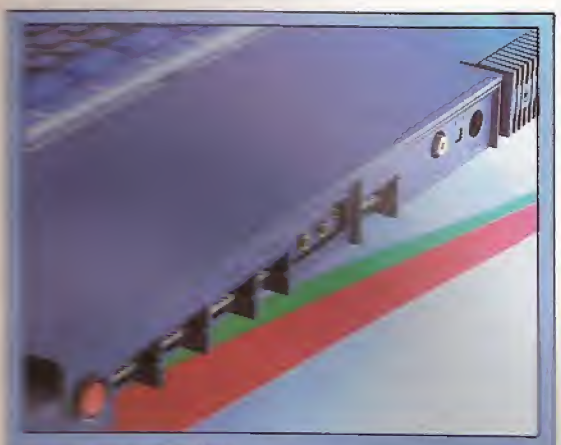
LENGUAJES DISPONIBLES
BASIC

TECLADO
69 teclas tipo máquina de escribir, incluyendo ocho teclas de función. Palanca de mando incorporada

DOCUMENTACION
Se incluye con el ordenador un breve manual de introducción para instalar el ordenador, y diagramas que explican el teclado y las interfaces. Hay igualmente un manual de aprendizaje de BASIC

VENTAJAS
El ordenador posee una buena versión de BASIC y excelentes capacidades para gráficos y sonido, y tiene un procesador de textos incorporado

DESVENTAJAS
En general, el estándar del hardware no está en consonancia con el del software. El teclado y la palanca de mando incorporada parecen insustanciales. La conexión de interfaces no estandarizadas significa que, al menos por ahora, los usuarios no podrán adquirir libremente sus periféricos



Interfaces para periféricos
El riesgo mayor del Enterprise es que posee conectores marginales como interfaces para periféricos, en vez de los convencionales tipo DIN. Aunque ello comporta que la gama de periféricos que se pueden conectar es más amplia, en la práctica muchos usuarios tendrán que acudir probablemente a los proveedores independientes para conseguir las interfaces adecuadas para las conexiones más populares

Cartucho de BASIC
A diferencia de la mayoría de las máquinas modernas, el BASIC (de Intelligent Software) se suministra en un cartucho de ROM de 16 K. Este cartucho se inserta en la ranura situada a uno de los lados de la máquina. El disponer de un lenguaje de programación basado en cartucho significa que el BASIC se podrá intercambiar fácilmente con otros lenguajes

Chris Stevens

Fiel al original

Veamos cómo diseñar para el Spectrum una interface que imite a la puerta para el usuario

El BBC Micro y el Commodore 64 incorporan una puerta para el usuario, para la conexión de una interface de propósito general. El robot que hemos construido y utilizado se conecta con el ordenador controlador a través de estas interfaces. Lamentablemente, el Spectrum no posee tal puerta para el usuario. En realidad, a excepción de una interface para cassette, no posee ninguna otra.

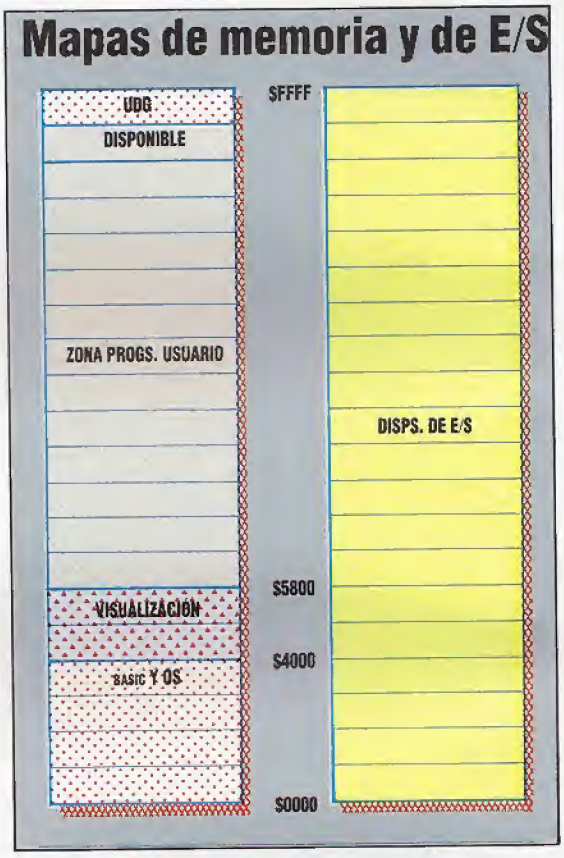
En cambio, el Spectrum se suministra con una puerta para ampliación, situada en la parte posterior de la máquina, que dispone de todas las conexiones internas: el bus de datos, el bus de direcciones, las líneas de control y otras varias señales, incluyendo las líneas de potencia. Estas diversas líneas se pueden utilizar para conectar a la máquina toda clase de interfaces, y es allí donde se puede construir una que imite a las puertas para el usuario.

Las puertas para el usuario se basan en un chip llamado VIA o adaptador para interface versátil (*versatile interface adaptor*). Éste es un chip complejo que se ocupa de numerosas tareas. El VIA proporciona al ordenador ocho líneas de entrada-/salida. Cada línea (cada bit del byte E/S) se puede configurar, mediante software, como entrada o salida. En el chip hay, además, diversas líneas de entrada y salida de control y un temporizador incorporado. Nuestra interface DIY para el Spectrum no es tan compleja, pero cumplirá las exigencias de nuestro robot. Se necesitan cuatro bits de salida y cuatro bits de entrada. Dado que la configuración in/out es estática (no es necesario cambiarla por software) no será necesario imitar la compleja disposición que tiene el RDD/Registro de Datos del chip VIA.

En la mayoría de los ordenadores basados en el microprocesador 6502 (como el BBC Micro y el Commodore 64), la puerta para el usuario y otras interfaces están conectadas a la circuitería de la máquina de la misma manera que un byte de memoria. Es decir, forman parte del mapa de memoria del ordenador. El Spectrum, sin embargo, utiliza un microprocesador Z80. Este chip posee un refinado sistema de conexión en interface incorporado. En vez de incluir los dispositivos de E/S en el mismo mapa de memoria que la RAM y la ROM, para tales dispositivos existe un mapa de memoria separado, paralelo al principal. Al mismo se accede desde BASIC con IN y OUT, de la misma forma en que se utilizan PEEK y POKE para acceder al mapa de memoria.

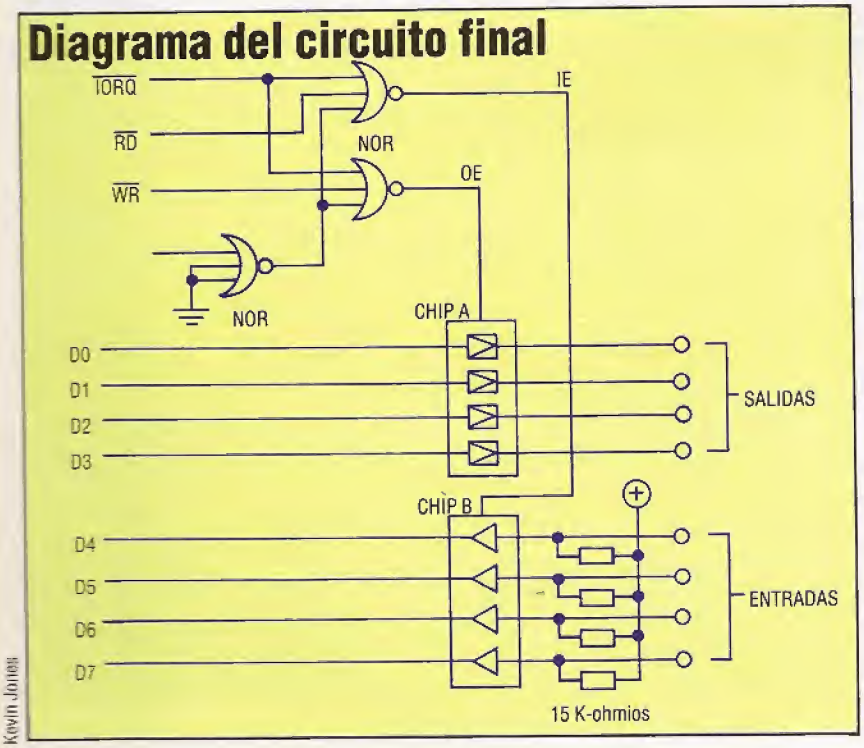
El Z80 distingue los dos mapas (memoria y E/S) gracias a un bit de dirección extra. Además de los

Mapas de memoria y de E/S
El robot se conecta en interface con el Spectrum designando una de las 65 535 posiciones en este mapa de E/S; las instrucciones IN y OUT nos permiten leer datos del mapa de E/S, o escribirlos en él, de la misma forma en que se utilizan PEEK y POKE con el mapa de memoria



Liz Dixon

Líneas lógicas
Combinando las líneas de solicitud de E/S, leer y escribir y del bit 5 de dirección a través de una serie de puertas NOR, podemos producir dos señales de salida. IE y OE habilitan los chips del buffer de entrada y salida de las cuatro líneas, permitiendo que las señales de control fluyan desde el ordenador hacia los motores paso a paso del robot y, por el contrario, que las señales provenientes de los sensores del robot entren para ser analizadas mediante software. La entrada se habilita cuando sólo está *high* WR la salida, sólo cuando está *high* RD



Kevin Jones

	A10	A8	RFSH	M1	-12v	+12v	WAIT	-5v	WR	RD	IORQ	MREQ	HALT	NMI	INT	D4	D3	D5	D6	D2	D1	D0			D7	A13	A15
A11	A9	BUSACK	ROMCS	A4	A5	A6	A7	RESET	BUSRQ	U	V	Y	VIDEO	0v	IORQGE	A3	A2	A1	A0	CK	0V	0V		9V	5V	A12	A14

16 bits que se emplean para direccionar los 64 K de memoria, otro bit controla si se está utilizando el mapa de memoria o el de E/S. Este bit extra está en forma de dos líneas de control provenientes del microprocesador. Éstas se denominan IORQ (*Input/Output ReQuest*) y MREQ (*Memory ReQuest*). Cuando se está direccionando una posición de memoria, la línea MREQ se pone *low*. Cuando se ha de direccionar un dispositivo de E/S, es la línea IORQ la que se pone *low*. En la puerta para ampliación del Spectrum están disponibles todas estas líneas, las 16 de dirección y las dos de control. Podemos, por consiguiente, diseñar una interface que responda sólo a ciertas direcciones y a la solicitud de E/S.

Algunas de las 65 535 posibles puertas de E/S ya se han utilizado en el Spectrum para la conexión interna en interface con el teclado, la interface para cassette y el zumbador. Las puertas de E/S que emplean las líneas de dirección de 0 a 4 son utilizadas por el Spectrum. Sólo haremos uso de la línea de dirección 5. Aunque ello significará que la interface "aparecerá" en muchos sitios del mapa de E/S (en cualquier dirección que utilice este bit), simplificará considerablemente la cuestión electrónica.

Nuestra interface posee datos tanto de entrada como de salida, de modo que deberemos emplear las líneas del Z80 para leer (RD) y escribir (WR) además de las líneas de dirección y de control. La interface ha de responder, entonces, al hecho de que las líneas IORQ y A5 se coloquen *low*, y al hecho de que también se coloquen *low* ya sea la línea RD como la WR. Cuando se satisfagan estas condiciones, la interface conectará el bus de datos del ordenador con las líneas de datos del robot. Ofrecemos la tabla de verdad para la interface.

Tabla de verdad

IE y OE son, respectivamente, las señales activadoras de las partes de entrada y salida de la interfase. El estado final de todas las líneas, incluyendo tanto a RD como a WR, no permite que estén todas *low* al mismo tiempo (el microprocesador no puede leer y escribir en una posición simultáneamente). La implementación de esta tabla de verdad es más sencilla en un circuito como el que indicamos.

Cuando se ponen *low* las líneas IORQ, RD y A5 (p. ej., cuando el microprocesador intenta leer de la puerta E/S 31), la línea IE de la interface se pone *high* y las cuatro conexiones de entrada se conectan a través de los cuatro bits *high* del bus de datos del ordenador a través del chip buffer A. Cuando las líneas IORQ y A5 y WR se ponen *low*, la OE se pone *high* y los cuatro bits *low* del bus de datos se conectan a las cuatro conexiones de salida de la interface. Cuando la línea OE se vuelve a poner *low*, cuando el procesador accede a otra dirección, el chip B retiene el patrón de salida hasta que se vuelve a acceder a él, proceso denominado *latching*.

Si en las cuatro líneas aparece alguna otra combinación, entonces las conexiones de interface se mantienen aisladas del bus de datos y no inciden ni en él ni en el ordenador. Este diseño produce la interface deseada para el robot en tres chips, uno que contiene las tres puertas OR de triple entrada (más una cuarta sin utilizar), el chip buffer (chip A) y el cerrojo (*latch*) (chip B).

Tabla de verdad de la interface

IORQ	WR	RD	A5	IE	OE
0	0	0	0	—	—
0	0	0	1	—	—
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	—	—
1	0	0	1	—	—
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Decir la verdad
La tabla de verdad muestra las señales requeridas a producir en las líneas habilitadoras de entrada y salida, para todas las combinaciones posibles de las cuatro señales de entrada. El caso en el cual WR y RD son ambas cero no puede darse en la práctica, dado que es imposible leer y escribir al mismo tiempo. Necesitamos que la entrada se habilite cuando IORQ, RD y A5 sean todas cero. IE, por consiguiente, se puede producir haciendo pasar estas tres líneas a través de una puerta NOR. Del mismo modo, se debe habilitar la salida cuando IORQ, WR y A5 estén *low*. Una segunda puerta NOR con estas tres entradas producirá OE

Horizontes ampliados
La puerta para ampliación situada en la parte posterior del Spectrum permite la conexión de dispositivos externos con la propia electrónica interna del ordenador. Están disponibles las 16 líneas de dirección y las 8 líneas de datos, junto con conexiones de patillas del procesador, como RD, WR y IORQ. Accediendo a estas patillas junto con las alimentaciones de voltaje disponibles, podemos conectar nuestro robot en interface con el Spectrum a través de la puerta para ampliación

La interface en funcionamiento
El Commodore 64 y el BBC Micro poseen chips de E/S especiales que nos permiten conectar el robot directamente al ordenador. Para utilizarlo con el Spectrum, sin embargo, necesitamos diseñar nuestra propia y simple placa de interface, que se enchufa directamente en la puerta para ampliación del Spectrum. Habiendo construido la interface, estaremos en condiciones de controlar el robot mediante software

Liz Heaney

Ficheros y archivos

Vamos, por fin, a abordar los ficheros del BBC Micro, examinando detalladamente la llamada OSFILE

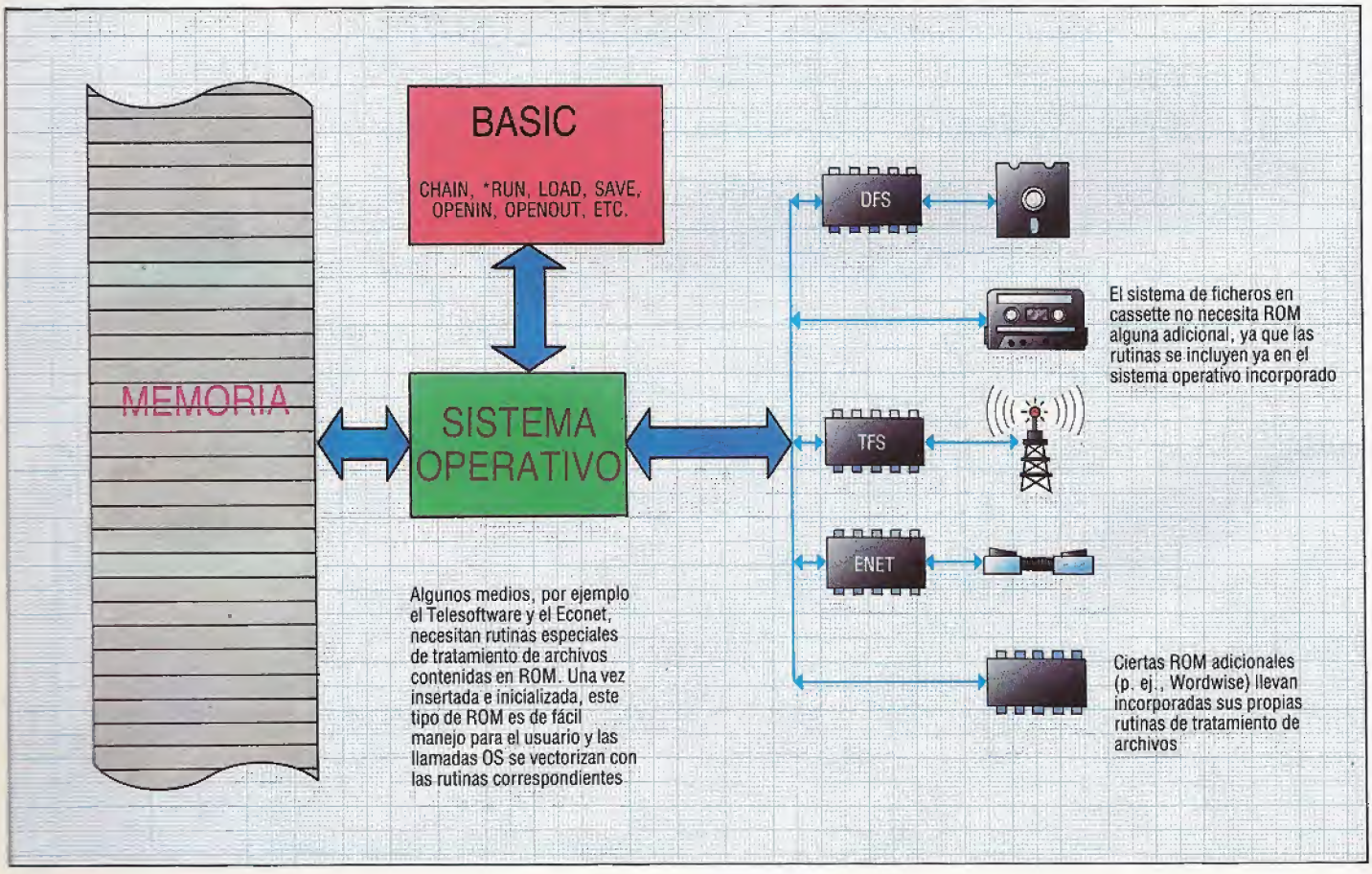
El sistema de ficheros de un ordenador se encarga de trasladar los datos de la memoria de éste hasta un medio de almacenamiento de larga estabilidad (almacén que puede ser una cassette, un disco *floppy* o flexible, etc.) y viceversa. El mismo BBC Micro es capaz de llevar a cabo el almacenamiento en los dos medios citados, pero es posible encontrar en el mercado otros dispositivos para ficheros tales como el sistema de red Econet, cartuchos de ROM y telesoftware. Aunque parezca que cada uno de estos dispositivos necesite una rutina de acceso diferente, y, por tanto, una llamada del OS diversa, la verdad es que el sistema de ficheros del BBC Micro se sirve siempre de las mismas rutinas del OS. Lo que no significa que todos los dispositivos funcionen igual. Son necesarias distintas ROM de sistema de archivos para cada medio, que tratan las distintas estructuras del fichero físico (la ordenación concreta de los bytes de datos en el almacén) y se comunican con la estructura de ficheros lógicos (registros, campos, etc.) mediante rutinas generales del OS.

El sistema de ficheros del BBC Micro cuenta con llamadas vectorizadas; al entrar en un sistema de archivos concreto damos permiso al OS para que se encargue de acomodar el nuevo sistema. Así, un programa que se escriba basado únicamente en rutinas oficiales del OS funcionará con cualquier sistema de almacenamiento. Naturalmente, puede que algunas operaciones (tales como la instrucción SAVE dada a una ROM) no sean admisibles con un determinado sistema de archivos.

Existen numerosas maneras de entrar un sistema de archivo. Examinaremos las más inmediatas:

- Si pulsamos Reset, para asegurar que se entra el sistema de archivos automáticamente, posicionaremos el chip de ROM con el sistema adecuado para que se convierta en el chip de ROM por defecto. Por lo general, con más de una ROM para sistemas de archivos será siempre seleccionada la que se encuentra a la izquierda inmediata de la ROM para el BASIC, suponiendo que éste se halla a la extrema derecha de los enchufes de ROM paginada.
- Se puede emplear BREAK junto con otra tecla. Por ejemplo, suponiendo que la máquina tiene incorporada la ROM adecuada, si pulsamos la tecla N junto con BREAK entraremos al sistema de archivos Econet.

Armario de archivo
El sistema de archivos proporciona un conjunto de rutinas para transferir datos entre la memoria, dispositivos de almacenamiento de larga duración y otros medios (inclusive el Telesoftware y las redes de usuario)



Kevin Jones

• Se puede también emplear una instrucción *. La instrucción *TAPE 12 entrará en el sistema para cassette de 1 200 baudios; la *DISC, en el sistema de ficheros en disco; *TELESOFT en el sistema Telesoftware y *NET en el sistema Econet. Naturalmente estas últimas tres instrucciones exigen que la máquina tenga conectada la ROM adecuada a ese sistema de archivo.

Empleamos el sistema de archivo virtualmente cada vez que encendemos la máquina, aunque sea sólo para cargar (LOAD) o guardar (SAVE) un programa. Estas dos instrucciones del BASIC hacen uso de las rutinas OS del sistema de archivos, al igual que CHAIN, OPENUP, OPENOUT, OPENIN, INPUT, PRINT y otras. También otras muchas instrucciones * emplean las rutinas OS del sistema de fichero. Son las que ilustramos en el cuadro. Pero hay otras instrucciones * que vienen provistas con ROM individuales para sistema de ficheros.

Antes de iniciar nuestro examen de algunas rutinas ROM específicas relacionadas con operaciones sobre ficheros en el BBC, es necesario aclarar que no pretendemos describir todos los usos de cada llamada que examinamos. Nos limitaremos a describir los usos más representativos de cada rutina OS.

Hay sólo siete rutinas OS relacionadas con ficheros y un par de llamadas OSWORD y OSBYTE. Las llamadas específicas son OSFILE, OSARGS, OSBGET, OSBPUT, OSFSC, OSFIND y OSGBPB.

La llamada OSFILE

Opera con ficheros completos de datos, lo cual constituye un buen lugar para empezar. Se emplea mediante operaciones del tipo SAVE, LOAD, *SAVE y otras que exigen operar con grandes bloques de datos. Estos archivos tienen bloques de información asociados, llamados *atributos*. Los atributos de un fichero incluyen información tal como LOAD ADDRESS, EXECUTION ADDRESS y longitud del archivo. La OSFILE posibilita al programador el manejo de estos atributos si se usa con ciertos sistemas de ficheros, como los discos. La OSFILE, como la OSWORD, requiere un bloque de parámetros (cuya dirección se pasa a la rutina con los registros X e Y, igual que para OSWORD). También vectorizada, se puede llamar con la dirección &FFDD. La operación concreta llevada a cabo por la rutina OSFILE se especifica en el registro A en el momento de hacerse la llamada. Hemos confeccionado dos cuadros: uno para mostrar la estructura del bloque de parámetros de OSFILE y otro para ilustrar las operaciones que realiza.

Cuando las operaciones se hacen en un BBC Micro estándar, que sólo necesita una dirección de 16 bits para especificar las posiciones de memoria, sólo son significativos, en los números de cuatro bytes que especifican las direcciones, el BmnS (Byte menos Significativo) y el byte siguiente. Si se observa el cuadro relativo a las operaciones, es fácil deducir que el sistema de ficheros en cassette no tiene muchas opciones disponibles desde OSFILE. Resulta claro que su principal utilización se refiere al sistema de ficheros en disco. La información del atributo del fichero depende del sistema de fichero que se emplee. Para un sistema de disco es muy útil, por ejemplo y dado que nos permite la modificación, la dirección de carga de un fichero mediante la sencilla alteración de la entrada correspondiente dentro del bloque de parámetros y realizando la llamada OSFILE con el valor en A adecuado.

Un ejemplo del empleo de OSFILE es el que se muestra en el siguiente programa para guardar el empleo mismo de OSFILE:

```
10 DIM bloque 20
20 DIM nombre 10
```

Bloque de parámetros para OSFILE

Registros X e Y con el desplazamiento desde dir.	Descripción
0	BmnS de la dir. del nbre. fichero
1	BmsS de la dir. del nbre. fichero
2	BmnS de la dir. de carga fichero
3	Sin especificar
4	Sin especificar
5	BmsS de la dir. de carga fichero
6	BmnS de la dir. de ejecución del archivo
7	Sin especificar
8	Sin especificar
9	BmsS de la dir. de ejecución del archivo
10	BmnS de datos a ser escritos en el fichero en una oper. de escrit., o dir. de destino en una de lect.
11	Sin especificar
12	Sin especificar
13	BmsS de la dir. de los datos
14	BmnS de (dir. final de datos+1) en una operación de escritura, o información del atributo del archivo por leer
15	Sin especificar
16	Sin especificar
17	BmsS de la dir. de datos

Operación realizada por llamadas OSFILE

Contenido de A	Descripción de la operación realizada	
	En disco	En cassette
0	Guarda un área de memoria	Guarda un área de memoria
1	Escribe en el catálogo	Carga el fich.
2	Escribe la dir. de carga	Carga el fich.
3	Escribe la dir. de ejecuc.	Carga el fich.
4	Escribe los atrib. del archivo	Carga el fich.
5	Lee el catálogo	Carga el fich.
6	Borra el fichero	Carga el fich.
255	Carga el fichero	Carga el fich.

```
30 BASIC=&8023:REM sirve para el BASIC II, para
   el BASIC I será $801F
40 X%=bloque MOD 256: REM apunta al bloque de
   parametros
50 Y%=bloque DIV 256
60 A%=0:REM especifica una operacion de
   escritura
70 bloque!0=nombre:REM direccion del nombre
   del archivo
80 bloque!2=PAGINA:REM la direccion de carga
   del programa
90                    :REM se pasara a PAGINA
100 bloque!6=BASIC:REM la direccion de ejecucion
    sera
110                   :REM el punto de entrada para
    BASIC
120 bloque!10=PAGINA:REM primer byte por
    escribir
130                   :REM es PAGINA
140 bloque!14=(TOP+1):REM fin de datos + 1
150 $nombre="FRED":REM poner el nombre de
    archivo legal
160 CALL &FFDD: REM ejecutarlo
170 END
```

El dar de baja un archivo en disco con OSFILE es de suma sencillez: basta con introducir en la memoria el nombre del archivo, hacer que las dos primeras entradas del bloque de parámetros indiquen el nombre, poner el registro A a 6 y llamar a OSFILE. Después de llamar a OSFILE con A=5, los bytes 2 y 5 del bloque retendrán la dirección de carga del fichero mencionado, y los bytes 6 y 9 contendrán la dirección de ejecución. Si el nombre del fichero no se logra encontrar, A abandonará la llamada con el valor 200. Si no hay fichero, A retendrá el valor 1.

Instr.* que emplean rutinas OS del sist. de archivos

Instrucción*	Descripción
*CAT, o bien, *	Genera un catálogo de ficheros disponibles en ese momento dentro del sistema de ficheros
*SAVE	Guarda en algún medio de ficheros un bloque de bytes de la memoria. Por ejemplo: *SAVE "nombre-fichero" inicio fin ejecución donde "inicio" es la dirección en hexa del primer byte por guardar, "fin" la del último byte+1 y "ejecución" la dirección de ejecución del fichero de bytes resultante. Nótese: "ejecución" es un parámetro opcional, y si se omite, la dirección de ejecución coincidirá con la de inicio; "fin" puede ser sustituido por "+longitud" donde "longitud" representa el número de bytes que se han de guardar
*LOAD	Carga un fichero de bytes a una dirección concreta en la memoria del ordenador. La orden se emplea así: *LOAD "nombre-fichero" dirección donde "dirección" es la dir. en hexa a donde han de cargarse los bytes. Si este parámetro se omite, los bytes se cargan en la dirección de donde fueron tomados para ser guardados
*RUN	Es como una versión en código máquina de CHAIN del BASIC. *RUN "nombre-fichero" cargará el fichero de bytes en la dir. donde se tomaron para ser guardados, ejecutándolos luego como un progr. en cód. máq., empezando por la dir. de ejecución guardada con los bytes
*SPOOL y *EXEC	Ya se estudiaron anteriormente
*OPT 1,x	Indica al OS el tipo de mensajes a imprimir durante las operaciones de los sist. de ficheros. Si x=0, no hay que imprimir mensajes; x=2 los proporciona breves y x=2, más largos
*OPT 2,x	Controla la acción del OS en caso de error durante una operación del sist. de fichero. Si x=0 se ignoran los errores, x=1 invita al usuario a insistir una vez hallado el error, y x=3 provoca la conclusión del trabajo del sist. en caso de hallar uno
*OPT 3,x	Establece el retardo temporal entre los bloques de datos que el sist. ha de escribir en cinta; "x" se mide en unidades de 0,1 s
*OPT 4,x	Sólo utilizada en sistemas para discos, se incluye aquí por su gran utilidad. Establece las opciones SHIFT-BREAK en disco. Véase el manual DFS para más detalles

El bosque encantado

Finalizamos nuestro proyecto de programación con un listado completo de "El bosque encantado"

El bosque encantado es un corto juego de aventuras diseñado para demostrar algunas de las técnicas de programación básicas de la programación de juegos de aventuras. Por necesidad, el juego es limitado tanto en su línea argumental como en la cantidad de opciones que se ofrecen al jugador. La construcción de *El bosque encantado* ilustra cómo se puede construir un programa de juego de aventuras alrededor de un esquema sencillo, cuya base es un mapa que refleja los diferentes escenarios y sus relaciones entre sí dentro del mundo de la aventura. Tanto si el juego tiene 16 escenarios como si tiene 600, debe ser diseñado utilizando el mapa como base. En éste se deben marcar, asimismo, los objetos y los "peligros" especiales. Obviamente, es importante que la trama argumental y los principales peligros se hayan elaborado con cierto detalle antes de comenzar con la programación. A menudo, la mejor forma de racionalizar y plasmar ideas para las reglas del juego es dibujar el mapa del mundo imaginario en un papel cuadriculado. Una vez que se ha llegado al concepto final, se puede comenzar la programación.

Por lo general la primera tarea consiste en recodificar el mapa gráfico bidimensional en forma de elementos de matriz. En el sistema utilizado en *El bosque encantado*, los datos de los escenarios están almacenados en dos matrices unidimensionales que retienen una descripción del escenario y una lista codificada de las salidas existentes de ese escenario. Los objetos presentes en el juego están retenidos en una matriz bidimensional, indicando una descripción del objeto y el escenario en el cual se halla. En esta etapa, tenemos una sencilla base de datos que representa el mundo de la aventura, y se pueden desarrollar rutinas simples de manipulación que le permitan al jugador desplazarse a través de los diversos escenarios y recoger o abandonar objetos. En esta etapa, las reglas específicas que regulan la forma en que se participa en el juego están en gran parte sin definir, exceptuando el equipar el juego con los objetos necesarios para que el jugador pueda sortear los peligros que lo acecharán en los escenarios especiales.

Esta versión del programa se escribió para el Commodore 64, pero funcionará en la mayoría de las máquinas que soporten versiones de BASIC tipo Microsoft. Quizá sean necesarias algunas modificaciones mínimas relacionadas con la generación de números aleatorios y el borrado de la pantalla. Asimismo, ofrecemos detallados *Complementos al BASIC* para el BBC Micro. Sin embargo, debido a la peculiar manipulación de matrices de variables en serie del Spectrum, los complementos para el Spectrum son demasiado extensos como para listarlos aquí. Habrá de remitirse a los que hemos ido ofreciendo a lo largo del proyecto. Para ayudarle con los complementos para el Spectrum, incluimos una tabla de conversión para los nombres de las variables en serie, que se utilizan en todos los complementos para el Spectrum que hemos ofrecido previamente.

Debido a que un juego de aventuras se basa en que el jugador descubra peligros y procedimientos ocultos con el objeto de recorrer el mundo de la aventura, un análisis detallado de cómo está construido el juego descubre inevitablemente los secretos del mismo. Al mostrar toda la dinámica interna de *El bosque encantado*, hemos, por lo tanto, debido revelarle los detalles del escenario del juego. *Digitaya* se fue ofreciendo a través de la serie junto con *El bosque encantado*, para ofrecerle una aventura con la cual jugar sin conocer de antemano el escenario. En el próximo capítulo le proporcionaremos el listado de *Digitaya*.

Tabla de conversión para el Spectrum

Microsoft	Spectrum	Finalidad de las variables
LN$()	L$()	Descripción de escenarios
EX$()	E$()	Salidas
IC$()	I$()	Objetos que lleva el jugador
IV$(,)	V$(,)	Objetos del juego
SN$	S$	Retiene frase a formatear
OW$	O$	"Palabra vieja" de rut. de formateo
NW$	N$	"Palabra nueva" de rut. de formateo
EX$	X$	Salidas del escenario actual
IS$	I$	Instrucción en curso
DR$	D$	Dir. especificada por la instr.
NN$	R$	Parte no verbo de instrucción
VB$	B$	Parte verbo de instrucción
CD$	C$	Palabra código

Complementos al BASIC

BBC Micro:
En el listado de *El bosque encantado* se deben sustituir las siguientes líneas:

```
207 RND(-TIME)
210 P=RND(10)

1160 CLS

4190 REPEAT:A$=GET$:UNTIL A$="S" OR A$="N"
4535 CR=RND(3)

6067 CC=0
```

El listado final

```
130 REM **       BOSQUE       **
140 REM **     ENCANTADO      **
180 :
200 GOSUB6000:REM LEER DATOS MATRIZ
205 GOSUB1000:REM HISTORIA HASTA AHORA
207 R=RND(-1)
210 P=INT(RND(TI)*10+1):REM PUNTO DE PARTIDA
220 :
230 REM **** AQUI EMPIEZA EL BUCLE PRINCIPAL ****
240 MF=0:REM BANDERA MOVIMIENTO
245 PRINT
250 GOSUB2000:REM DESCRIBIR ESCENARIO
255 GOSUB2300:REM DESCRIBIR SALIDAS
257 GOSUB2700:REM ES ESPECIAL P?
258 IF SF=1 THEN 300:REM SIGUIENTE INSTRUCCION
260 PRINT:INPUT"INSTRUCCIONES";IS$
270 GOSUB2500:REM DESCOMPONER INSTRUCCION
275 IF F=0 THEN 260:REM INSTRUCCION NO VALIDA
280 GOSUB3000:REM INSTRUCCIONES NORMALES
290 IF VF=0 THENPRINT:PRINT"NO COMPRENDO"
300 IF MF=1 THEN 240:REM NUEVO ESCENARIO
310 IF MF=0 THEN 260:REM NUEVA INSTRUCCION
320 :
390 END
1000 REM **** S/R HISTORIA HASTA AHORA ****
1010 SN$="BIENVENIDO AL BOSQUE ENCANTADO"
1020 GOSUB5500:REM FORMATEAR
1030 PRINT
1040 SN$="AL DESPERTARTE DE UN PROFUNDO SUEÑO, EL "
1050 SN$=SN$+"SUELO DEL BOSQUE ESTA SUAVE Y SECO. "
1060 SN$="NO SABES COMO HAS LLEGADO HASTA AQUI "
1070 SN$=SN$+"PERO SABES QUE DEBES ENCONTRAR EL "
1080 SN$=SN$+"POBLADO QUE LINDA CON EL BOSQUE PARA "
1090 SN$=SN$+"ESTAR A SALVO."
1100 GOSUB5500:REM FORMATEAR
1110 PRINT
1120 SN$="MIRAS A TU ALREDEDOR, TRATANDO DE ORIENTARTE."
1130 GOSUB6500:REM FORMATEAR
1140 PRINT:PRINT"PULSA CUALQUIER TECLA PARA EMPEZAR"
1150 GET A$:IF A$="" THEN 1150
1160 PRINTCHR$(147):REM LIMPIAR PANTALLA
1170 RETURN
1180 :
2000 REM **** DESCRIBIR ESCENARIO ****
2010 SN$="TE HALLAS "+ES$(P):GOSUB5500
2020 SN$="VES "
2030 REM ** VERIFICAR INVENTARIO PARA OBJETO **
2040 F=0:SP$=" "
2050 FOR I=1 TO 3
2060 IF VAL(IV$(I,2))<>P THEN 2080
2070 SN$=SN$+SP$+"UNA "+IV$(I,1):F=1:SP$=", "
2080 NEXT I
2090 IF F=0 THEN SN$=SN$+"NINGUN OBJETO"
2100 GOSUB5500:REM FORMATEAR
2110 RETURN
2120 :
2299 :
2300 REM **** S/R DESCRIBIR SALIDAS ****
2310 SL$=SL$(P)
2320 NR=VAL(LEFT$(SL$,2))
2330 ES=VAL(MID$(SL$,3,2))
2340 SU=VAL(MID$(SL$,5,2))
2350 OE=VAL(RIGHT$(SL$,2))
2353 :
2355 IF(NR OR ES OR SU OR OE)=0 THEN RETURN
2360 PRINT:SN$="HAY SALIDAS POR EL "
2370 IF NR<>0 THEN SN$=SN$+"NORTE "
2380 IF ES<>0 THEN SN$=SN$+"ESTE "
2390 IF SU<>0 THEN SN$=SN$+"SUR "
2400 IF OE<>0 THEN SN$=SN$+"OESTE "
2410 GOSUB5500:REM FORMATEAR
2415 PRINT
2420 RETURN
2430 :
2500 REM **** S/R DESCOMPONER INSTRUCCION ****
2510 IF IS$="LISTAR" OR IS$="FIN" THEN VB$=IS$:F=1:RETURN
2515 IF IS$="MIRAR" THEN VB$=IS$:F=1:RETURN
2520 F=0
2530 LS=LEN(IS$)
2540 FOR C=1 TO LS
2550 A$=MID$(IS$,C,1)
2560 IF A$<>" " THEN 2590
2570 VB$=LEFT$(IS$,C-1):F=1
2580 NN$=RIGHT$(IS$,LS-C):C=LS
2590 NEXT C
2600 :
2610 IF F=1 THEN RETURN
2620 PRINT:PRINT"NECESITO AL MENOS DOS PALABRAS"
2630 RETURN
2650 :
2700 REM **** S/R ES ESPECIAL P ****
2705 SF=0:REM QUITAR BANDERA ESPECIAL
2707 REM ** FANTASMA AL AZAR **
2710 IF P>4 AND RND(1)<0.1 THEN GOSUB 4290:RETURN
2715 :
2716 REM ** OTROS ESCENARIOS ESPECIALES **
2720 ON P GOSUB4590,4870,5100,4590
2730 RETURN
2735 :
3000 REM **** S/R INSTRUCCIONES NORMALES ****
3010 VF=0:REM BANDERA DE VERBO
3020 IF VB$="AVANZAR" OR VB$="IR" THEN VF=1:GOSUB3500
3030 IF VB$="RECOGER" OR VB$="COGER" THEN VF=1:GOSUB3700
3040 IF VB$="DEJAR" OR VB$="PONER" THEN VF=1:GOSUB3900
3050 IF VB$="LISTAR" OR VB$="INVENTARIO" THEN VF=1:GOSUB4100
3055 IF VB$="MIRAR" THEN VF=1:MF=1:RETURN
3060 IF VB$="FIN" OR VB$="TERMINAR" THEN VF=1:GOSUB4170
3070 RETURN
3080 :
3500 REM **** S/R MOVER ****
3505 GOSUB3630:REM BUSCAR DIRECCION
3510 MF=1:REM ESTABLECER BANDERA DE MOVIMIENTO
3520 DR$=LEFT$(NN$,1)
3530 IF DR$<>"N" AND DR$<>"E" ANDDR$<>"S" AND DR$<>"O"THENGOTO3590
3540 IF DR$="N"AND NR<>0 THENP=NR:RETURN
3550 IF DR$="E"AND ES<>0 THENP=ES:RETURN
3560 IF DR$="S"AND SU<>0 THENP=SU:RETURN
3570 IF DR$="O"AND OE<>0 THENP=OE:RETURN
3580 PRINT:PRINT"NO PUEDES ";IS$
3585 MF=0:RETURN
3590 REM ** EL NOMBRE NO ES UNA DIRECCION **
3600 PRINT"QUE ES ";NN$;" ?"
3610 MF=0:RETURN
3620 :
3630 REM **** S/R BUSCAR DIRECCION ****
3640 NN$=NN$+" ":LN=LEN(NN$):C=1
3645 FOR I=1 TO LN
3650 IF MID$(NN$,I,1)<>" " THEN NEXT I:RETURN
3655 W$=MID$(NN$,C,I-C):C=I+1
3660 IF W$="NORTE" OR W$="ESTE" THEN NN$=W$:I=LN
3665 IF W$="SUR" OR W$="OESTE" THEN NN$=W$:I=LN
3670 NEXT I
3675 RETURN
3700 REM ** S/R RECOGER **
3710 GOSUB 5300:REM ES VALIDO EL OBJETO
3720 IF F=0 THEN SN$="NO HAY NINGUNA "+W$:GOSUB5500:RETURN
3730 OV=F:GOSUB5450:REM VERIFICAR INVENTARIO
3740 IF HF=1 THEN SN$="YA TIENES LA "+IV$(F,1):GOSUB5500:RETURN
3750 :
3755 REM ** ESTA AQUI EL OBJ ? **
3760 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+" NO ESTA AQUI":GOSUB5500:RETURN
3770 :
3780 REM ** AÑADIR OBJETO A LA LISTA **
3790 A=0
3800 FOR J=1 TO 2
3810 IF IC$(J)=" " THEN IC$(J)=IV$(F,1):AF=1:J=2
3820 NEXT J
3830 :
3840 REM ** CUOTA CUBIERTA **
3850 IF AF=0 THEN PRINT"YA TIENES DOS OBJETOS":RETURN
3860 :
3870 SN$="RECOGES LA "+IV$(F,1):GOSUB5500
3880 IV$(F,2)="-1":REM ELIMINAR ENTRADA EN INVENTARIO
3890 RETURN
3895 :
3900 REM **** S/R DEJAR ****
3910 GOSUB5300:REM OBJETO VALIDO
3920 IF F=0 THEN SN$="NO HAY NINGUNA "+W$:GOSUB5500:RETURN
3930 :
3940 REM ** ESTA OBJ EN INVENTARIO OBJ TRANSPORTADOS **
3950 OV=F:GOSUB5450
3960 IF HF=0 THEN SN$="TU NO TIENES LA "+IV$(F,1):GOSUB5500:RETURN
3970 :
3980 REM **** DEJAR OBJETO ****
3990 SN$="DEJAS LA "+IV$(F,1):GOSUB5500
4000 IV$(F,2)=STR$(P):REM HACER ENTRADA EN INVENTARIO
4010 :
4020 REM ** SUPRIMIR OBJ DEL INVENTARIO OBJ TRANSPORTADOS **
4030 FOR J=1TO2
4040 IF IC$(J)=IV$(F,1) THEN IC$(J)=" ":J=2
4050 NEXT J
4060 RETURN
4070 :
4100 REM **** LISTAR INVENTARIO OBJ TRANSPORTADOS ****
4110 PRINT"OBJETOS QUE LLEVAS CONTIGO:"
4120 FOR I=1 TO 2
4130 PRINT"   ";IC$(I)
4140 NEXT I
4150 RETURN
4160 :
4170 REM **** S/R FIN DEL JUEGO ****
4180 PRINT:PRINT"ESTAS SEGURO (S/N) ?"
4190 GET A$:IF A$<>"S" AND A$<>"N" THEN 4190
4200 IF A$="N" THEN RETURN
4210 END
4220 :
4290 REM **** S/R FANTASMA AL AZAR ****
4295 SF=1:GC=0
4300 SN$="SIENTES UN ESCALOFRIO QUE TE RECORRE LA COLUMNA"
4305 SN$=SN$+" VERTEBRAL. DE PRONTO SURGE DE DETRAS DE"
4310 SN$=SN$+" LOS ARBOLES UNA APARICION Y"
4315 SN$=SN$+" AVANZA HACIA TI":GOSUB5500:REM FORMATEAR
4320 :
4325 SN$="EL FANTASMA SE TE ACERCA MAS Y MAS":GOSUB5500
4330 GC=GC+1:IF GC>4 THEN GOSUB4455:REM
4335 PRINT:INPUT"INSTRUCCIONES";IS$
4340 GOSUB2500:REM DESCOMPONER INSTRUCCION
4345 IF F=0 THEN 4325:REM SIGUIENTE INSTRUCCION
4350 OP=P:GOSUB3000:REM ANALIZAR INSTRUCCION
4355 IF MF=1 AND VB$="AVANZAR" THEN GOSUB4400:GOTO 4325
4357 IF MF=1 AND VB$="MIRAR" THEN GOSUB2000:GOSUB2300:GOTO4325
4360 IF VF=1 THEN 4325:REM SIGUIENTE INSTRUCCION
4365 REM ** NUEVAS PALABRAS DE INSTRUCCION **
4370 IF VB$="MATAR" OR VB$="LUCHAR" THEN GOSUB4425:GOTO 4325
4375 :
4385 IF VB$="CANTAR" THEN GOSUB4500:RETURN
4390 SN$="NO COMPRENDO":GOSUB5500:GOTO4325
4395 :
4400 REM ** INTENTAR MOVERSE**
4405 SN$="ESTAS PARALIZADO POR EL TERROR Y NO PUEDES"
4410 SN$=SN$+" MOVERTE... TODAVIA":MF=0:GOSUB5500:P=OP
4415 RETURN
4420 :
4425 REM ** LUCHAR O MATAR **
4430 SN$="EL FANTASMA ES UN SER SOBRENATURAL"
4435 SN$=SN$+" Y SE RIE DE TUS TIBIOS INTENTOS"
4440 SN$=SN$+" POR HACERLE DAÑO":GOSUB5500
4445 RETURN
4450 :
4455 REM ** MUERTE **
4460 SN$="EL DOLOR QUE SIENTES EN EL PECHO SE VUELVE INSOPORTABLE"
4465 SN$=SN$+" Y TE DESPLOMAS SOBRE EL PISO LLENO DE HOJAS DEL BOSQUE.":GOSUB5500
4470 SN$="TU ESPIRITU SE ELEVA DE TU CUERPO INERTE"
4475 SN$=SN$+" Y FLOTAS ENTRE LA NIEBLA PARA UNIRTE"
```

Kevin Jones

```
4480 SN$=SN$+" A LAS OTRAS ALMAS ATORMENTADAS DE"
4485 SN$=SN$+" EL BOSQUE ENCANTADO.":GOSUB5500
4490 END
4495 :
4500 REM ** CANTAR **
4505 SN$="SABES TRES CANCIONES. CUAL DE ELLAS ELEGIRIAS ?":GOSUB5500
4510 SN$="1) EL TEMA DE LOS CAZAFANTASMAS":GOSUB5500
4515 SN$="2) THERE'S A GHOST IN MY HOUSE":GOSUB5500
4520 SN$="3) WAY DOWN UPON THE SWANEE RIVER":GOSUB5500
4525 PRINT:INPUT"HAZ TU ELECCION";C$
4530 IF VAL(C$)> 3 OR VAL(C$)< 1 THEN PRINT:PRINT"NO VALE":GOTO4525
4535 CR=INT(RND(1)*3)+1
4537 IF CR<>VAL(C$) THEN GOSUB4542:REM CANCION EQUIVOCADA
4540 GOSUB4565:RETURN:REM CORRECTA
4542 REM **** S/R MELODIA EQUIVOCADA ****
4545 SN$="EL FANTASMA TIENE ESPECIAL AVERSION POR"
4550 SN$=SN$+" ESA MELODIA Y SE ABALANZA SOBRE TI."GOSUB5500
4555 GOSUB4455:REM MUERTE
4560 :
4565 REM ** MELODIA CORRECTA **
4570 SN$="EL FANTASMA SE APACIGUA AL OIRTE INTERPRETAR LA MELODIA"
4575 SN$=SN$+" Y SE EVAPORA EN EL AIRE":GOSUB5500
4580 RETURN
4585 :
4590 REM **** S/R ENTRADA AL TUNEL ****
4600 SF=1
4605 SN$="HAS LLEGADO A LA BOCA DE UN GRAN TUNEL":GOSUB5500
4610 SN$="PUEDES ENTRAR EN EL TUNEL O RETROCEDER POR EL SENDERO":GOSUB5500
4620 :
4625 PRINT:INPUT"INSTRUCCIONES";IS$
4630 GOSUB2500:REM DESCOMPONER INSTRUCCION
4635 IF F=0 THEN 4625:REM INSTRUCCION NO VALIDA
4637 GOSUB3000:REM INSTRUCCIONES NORMALES
4640 IF MF=1 THEN RETURN:REM EL JUGADOR RETROCEDE
4645 IF VF=1 THEN 4625:REM INSTRUCCION OBEDECIDA
4650 IF VB$="ENTRAR" THEN GOSUB 4700:RETURN
4660 IF VB$="RETROCEDER" AND P=4 THEN MF=1:P=6:RETURN
4665 IF VB$="RETROCEDER" AND P=1 THEN MF=1:P=9:RETURN
4667 SN$="NO COMPRENDO":GOSUB5500:GOTO 4625
4700 REM ** ENTRAR EN EL TUNEL **
4705 SN$="ENTRAS EN EL TUNEL PERO ESTA DEMASIADO OSCURO COMO"
4710 SN$=SN$+" PARA ABRIRTE CAMINO.":GOSUB5500
4725 PRINT:INPUT"INSTRUCCIONES";IS$
4730 GOSUB2500:REM DESCOMPONER INSTRUCCION
4732 :
4735 IF F=0 THEN 4725:REM INSTRUCCION NO VALIDA
4740 OP=P:GOSUB3000:REM INSTRUCCIONES NORMALES
4745 IF MF=1 THEN SN$="ESTA TAN OSCURO QUE SOLO VES":P=OP
4747 IF MF=1 THENSN$=SN$+" LA ENTRADA AL TUNEL":GOSUB5500:MF=0:GOTO4725
4750 IF VF=1 THEN 4725:REM INSTRUCCION OBEDECIDA
4755 IF VB$="RETROCEDER" AND P=4 THEN MF=1:P=6:RETURN
4760 IF VB$="RETROCEDER" AND P=1 THEN MF=1:P=9:RETURN
4762 IF VB$<>"USAR" ANDVB$<>"ENCENDER"THEN SN$="NO COMPRENDO"
4765 IF VB$<>"USAR" ANDVB$<>"ENCENDER" THEN GOSUB5500:GOTO4725
4777 :
4780 REM ** BUSCAR LAMPARA **
4790 GOSUB5300:REM OBJETO VALIDO?
4795 OV=F:GOSUB5450:REM LLEVA CONSIGO EL OBJETO?
4797 IF F=0 THEN SN$="NO HAY NINGUNA "+W$:GOSUB5500:GOTO4725
4800 IF HF=0 THEN SN$="TU NO TIENES LA "+IV$(F,1):GOSUB5500:GOTO4725
4810 REM ** ES EL OBJ LA LAMPARA ? **
4815 IF F<>2 THEN SN$="LA "+IV$(F,1)+" NO SIRVE":GOSUB5500:GOTO4725
4825 REM ** EXITO **
4840 SN$="USAS LA LAMPARA PARA ILUMINARTE A TRAVES DEL TUNEL"
4845 SN$=SN$+" Y FINALMENTE SALES POR LA SALIDA.":GOSUB5500
4850 IF P=1 THEN MF=1:P=4:RETURN
4855 IF P=4 THEN MF=1:P=1:RETURN
4860 :
4870 REM **** S/R PANTANO ****
4875 SF=1
4880 SN$="EMPIEZAS A HUNDIRTE EN EL PANTANO.":GOSUB5500
4882 PRINT:INPUT"INSTRUCCIONES";IS$
4884 GOSUB2500:REM DESCOMPONER INSTRUCCION
4885 IF F=0 THEN 4885:REM NO VALIDA
4887 GOSUB3000:REM INSTRUCCIONES NORMALES
4888 IF VB$="NADAR" THENGOSUB2000:GOTO4885
4915 IF VB$="DEJAR" THEN IV$(F,2)="-2":REM OBJ PERDIDO PARA SIEMPRE
4917 IF VF=1 THEN 4885:REM INSTRUCCION NORMAL
4920 REM ** NUEVAS INSTRUCCIONES **
4925 IF VB$<>"NADAR" THEN SN$="NO COMPRENDO":GOSUB5500:GOTO4885
4930 REM ** NADAR **
4932 F=0
4935 FOR I=1 TO 2
4940 IF IC$(I)<>"" THEN F=F+1
4950 NEXT I
4955 IF F<2 THENGOSUB5800:RETURN:REM ALEJARSE NADANDO
4960 GOSUB 5000:RETURN:REM LLEVA CONSIGO DOS OBJ
5000 REM **** S/R LLEVA DOS OBJETOS ****
5010 SN$="LOS OBJETOS TE EMPUJAN HACIA ABAJO Y TE HUNDES.":GOSUB5500
5012 PRINT:INPUT"INSTRUCCIONES";IS$
5015 GOSUB2500:REM DESCOMPONER INSTRUCCION
5020 IF VB$<>"DEJAR" THENGOSUB5080:REM HUNDIR
5025 GOSUB3900:IV$(F,2)="-2":REM DEJAR OBJ
5030 IF HF=0 OR F=0 THEN 5080:REM HUNDIR
5035 REM **** ALEJARSE NADANDO ****
5040 SN$="AHORA PUEDES ATRAVESAR EL PANTANO A NADO. EN QUE DIRECCION
     IRAS?":GOSUB5500
5050 SL$(2)="00080605":GOSUB2300:REM DEFINIR Y VISUALIZAR SALIDAS
5055 PRINT:INPUT"INSTRUCCIONES";IS$
5060 GOSUB2500:REM DESCOMPONER INSTRUCCION
5062 IF F=0 THEN 5055:REM NO VALIDA
5065 GOSUB3500:REM MOVER
5067 SL$(2)="00000000":REM DATOS SALIDA CERO
5070 RETURN
5075 :
5080 REM **** S/R HUNDIR ****
5085 SN$="TE HUNDES EN EL PANTANO Y TE AHOGAS":GOSUB5500
5090 END
5100 REM **** S/R POBLADO ****
5102 SF=1
5105 SN$="EL POBLADO ESTA RODEADO POR UNA ALTA MURALLA.":GOSUB5500
5106 IF GF<>0 THEN GOSUB5190:RETURN:REM PUERTA
5107 SN$="EN LA PUERTA DE ACCESO AL POBLADO HAY UN GUARDIAN":GOSUB5500
5115 PRINT:INPUT"INSTRUCCIONES";IS$
5120 GOSUB2500:IF F=0 THEN5115:REM NO VALIDA
5125 GOSUB3000:REM INSTRUCCIONES NORMALES
5130 IF VB$="MIRAR"THENGOSUB2000:REM DESCRIBIR
5135 IF VB$="AVANZAR"AND MF=1 THEN RETURN
5140 IF VF=1 THEN 5115:REM SIGUIENTE INSTRUCCION
5145 IF VB$<>"MATAR" THEN SN$="NO COMPRENDO":GOSUB5500:GOTO5115
5150 REM **MATAR **
5155 SN$="QUE USARAS PARA MATAR AL GUARDIAN?":GOSUB5500
5160 SN$="ENTRAR OBJETO 0 <I> PARA INSTRUCCION":GOSUB5500
5162 INPUT IS$:IF IS$="I" THEN 5115
5165 GOSUB2500:REM DESCOMPONER
5167 IF F=0 THEN 5160:REM NO VALIDA
5170 GOSUB5300:IF F=0 THEN SN$ ="NO HAY NINGUNA "+W$:GOSUB5500:GOTO5160
5172 OV=F:GOSUB5450:REM LLEVA CONSIGO EL OBJ
5174 IF HF=0 THEN SN$="TU NO TIENES LA "+IV$(F,1):GOSUB5500:GOTO5160
5175 IF F<>1 THEN SN$="LA "+IV$(F,1)+" NO SIRVE":GOSUB5500:GOTO5160
5180 SN$="MATAS AL GUARDIAN":GOSUB5500:GF=1
5185 :
5190 REM **** S/R PUERTA CERRADA CON LLAVE ****
5195 SN$="TE ADELANTAS E INTENTAS ABRIR LA PUERTA DE ACCESO AL POBLADO"
5200 SN$=SN$+" PERO ESTA CERRADA Y NO SE MUEVE":GOSUB5500
5205 PRINT:INPUT"INSTRUCCIONES";IS$
5210 GOSUB2500:IF F=0 THEN 5205:REM NO VALIDA
5215 GOSUB3000:REM INSTRUCCIONES NORMALES
5220 IF VB$="MIRAR" THEN GOSUB2000:REM DESCRIBIR
5225 IF VB$="AVANZAR" AND MF=1 THEN RETURN
5230 IF VF=1 THEN 5205:REM SIGUIENTE INSTRUCCION
5232 IF VB$="USAR" THEN 5240
5234 IF VB$="ABRIR" THEN SN$="COMO?":GOSUB5500:GOTO5205
5235 SN$="NO COMPRENDO":GOSUB5500:GOTO5205
5240 GOSUB5300:REM OBJETO VALIDO
5242 OV=F:GOSUB5450:REM LLEVA CONSIGO EL OBJ
5244 IF F=0 THEN SN$="NO HAY NINGUNA "+W$:GOSUB5500:GOTO5205
5246 IF HF=0 THEN SN$="TU NO TIENES LA "+IV$(F,1):GOSUB5500:GOTO5205
5248 IF F<>3 THEN SN$="LA "+IV$(F,1)+" NO SIRVE":GOSUB5500:GOTO 5205
5250 REM ** ATRAVESAR LA PUERTA Y A SALVO **
5255 SN$="TU ABRES LA PUERTA Y, DISFRAZANDOTE CON LA"
5260 SN$=SN$+" ROPA DEL GUARDIAN MUERTO, ATRAVIESAS EL POBLADO SIN SER
     DESCUBIERTO."
5265 SN$=SN$+" Y LLEGAS A LA SEGURIDAD DEL MUNDO EXTERIOR.":GOSUB5500
5270 END
5299 :
5300 REM **** S/R OBJETO VALIDO ****
5310 NN$=NN$+" ":LN=LEN(NN$):C=1:F=0
5315 FOR K=1 TO LN
5320 IF MID$(NN$,K,1)<>" " THEN NEXT K:RETURN
5325 W$=MID$(NN$,C,K-C):C=K+1
5330 LW=LEN(W$)
5335 FOR J=1 TO3
5340 LI=LEN(IV$(J,1)):REM LONGITUD DEL OBJETO
5350 FOR I=1 TO LI-LW+1
5360 IF MID$(IV$(J,1),I,LW)=W$ THEN F=J:I=LI:J=3:K=LN
5370 NEXT I,J,K
5380 RETURN
5390 :
5450 REM **** S/R LLEVA OBJETO CONSIGO ****
5460 HF=0
5470 IF IV$(OV,2)="-1" THEN HF=1
5480 RETURN
5490 :
5500 REM **** S/R FORMATEAR SALIDA ****
5510 LC=0:      REM CONTADOR CAR/LINEA
5520 OC=1:      REM VALOR INICIAL CONTADOR
                ANTIGUO
5530 OW$=" ":  REM VALOR INICIAL PALABRA
                ANTIGUA
5540 LL=40:     REM LONGITUD LINEA
5550 SN$=SN$+" FICTICIO"
5560 PRINT
5570 FOR C=1 TO LEN (SN$)
5580 LC=LC+1
5590 IF MID$(SN$,C,1)=" " THEN GOSUB5800
5600 NEXT C
5605 PRINT
5610 RETURN
5620 :
5800 REM ** S/R VERIFICAR FINAL DE LINEA **
5810 NW$=MID$(SN$,OC,C-OC+1):REM PALABRA NUEVA
5820 IF LC<LL THEN PRINTOW$;:GOTO5840
5830 PRINTOW$:LC=LEN(NW$)
5840 OC=C+1:OW$=NW$
5850 RETURN
6000 REM **** LEER DATOS OBJ Y MAPA ****
6010 DIM IV$(3,2),ES$(10),SL$(10),IC$(2)
6020 FOR C=1 TO 3
6030 READ IV$(C,1),IV$(C,2)
6040 NEXT C
6050 :
6060 FOR C=1 TO 10
6065 READ ES$(C),SL$(C)
6070 CC=CC+VAL(SL$(C)):REM SUMA DE CONTROL TOTAL
6080 NEXT C
6090 :
6100 READ CD:IF CD<>CC THENPRINT"ERROR EN SUMA DE CONTROL":STOP
6110 :
6120 REM **** DATOS OBJETO ****
6130 DATA ESCOPETA,10,LAMPARA,9,LLAVE,5
6140 :
6150 REM **** DATOS MAPA ****
6160 DATA JUNTO A LA ENTRADA A UN TUNEL,00000900
6170 DATA EN UN PANTANO,00000000
6180 DATA EN UN POBLADO,07000000
6190 DATA JUNTO A LA ENTRADA A UN TUNEL,05060000
6200 DATA EN UN SENDERO,00020400
6210 DATA EN UN SENDERO,02070004
6220 DATA EN UN SENDERO,0800306
6230 DATA EN UN SENDERO,09000702
6240 DATA EN UN SENDERO,01100800
6250 DATA EN UN CLARO,00000009
6260 REM ** DATOS SUMA DE CONTROL **
6270 DATA 32253121
6280 RETURN
```

Paquete de película

"Ghostbusters" (Cazafantasmas) es un emocionante juego llegado de Estados Unidos e inspirado en la película del mismo nombre

Ghostbusters es una película que ha tenido un clamoroso éxito tanto en Estados Unidos como en Europa. Al lanzar el juego para ordenador del mismo nombre, los editores, Activision, sin duda alguna confiaron en que la magia volviera a funcionar otra vez.

El paquete se desarrolló en colaboración con la productora de la película, Columbia Pictures. Esta colaboración significó que los programadores pudieron seguir de cerca el guión cinematográfico sin el temor de ser demandados por los propietarios del copyright. El juego también utiliza la melodía del tema musical de la película, que en el verano de 1984 fue uno de los singles más vendidos en el mercado discográfico europeo.

Ghostbusters es la historia de tres investigadores universitarios que montan en Nueva York una agencia para cazar fantasmas y descubren que los duendes burlones están concentrándose en un edificio de apartamentos preparándose para el fin del mundo. El juego comienza dando dinero al jugador para que se lo gaste en un coche y en armas para "cazar fantasmas". Éstas incluyen un "detector de energía PK", para avisar de la proximidad de fantasmas, un "aspirador fantasmal", para succionar a los espíritus, y un sistema portátil de confinamiento láser, que le permite al jugador conducir a los fantasmas hasta una trampa especial. Una vez adquiridas las armas, el jugador debe aventurarse por las calles de Nueva York para tratar de ganarse la vida cazando fantasmas.

La escena cambia entonces a un mapa de la ciudad. En el centro de éste se halla el edificio en el cual se están concentrando los fantasmas, atraídos por el pérfido Zuul. Los fantasmas se muestran en diversos puntos de la ciudad y el jugador debe conducir el coche cazafantasmas hacia esos lugares. A medida que el vehículo va recorriendo las calles, aparecen en la pantalla fantasmas llamados *roamers* (vagabundos). Si el jugador ha equipado al coche con un "aspirador fantasmal", que se instala en el techo, podrá absorber *roamers* para conseguir puntos extras.

Si al llegar con el coche al lugar el fantasma todavía está allí, usted puede preparar el equipo cazafantasmas. Primero se coloca una trampa en el centro de la pantalla, y se puede conducir al fantasma hacia ella mediante los láseres. Cuando éste se halle directamente encima, la trampa se suelta y el fantasma cae en ella. El tema musical de *Ghostbusters*, que se oye constantemente como acompañamiento de fondo, se interrumpirá en este punto y una voz felicitará al jugador exclamando *"Ghostbuster!"*.

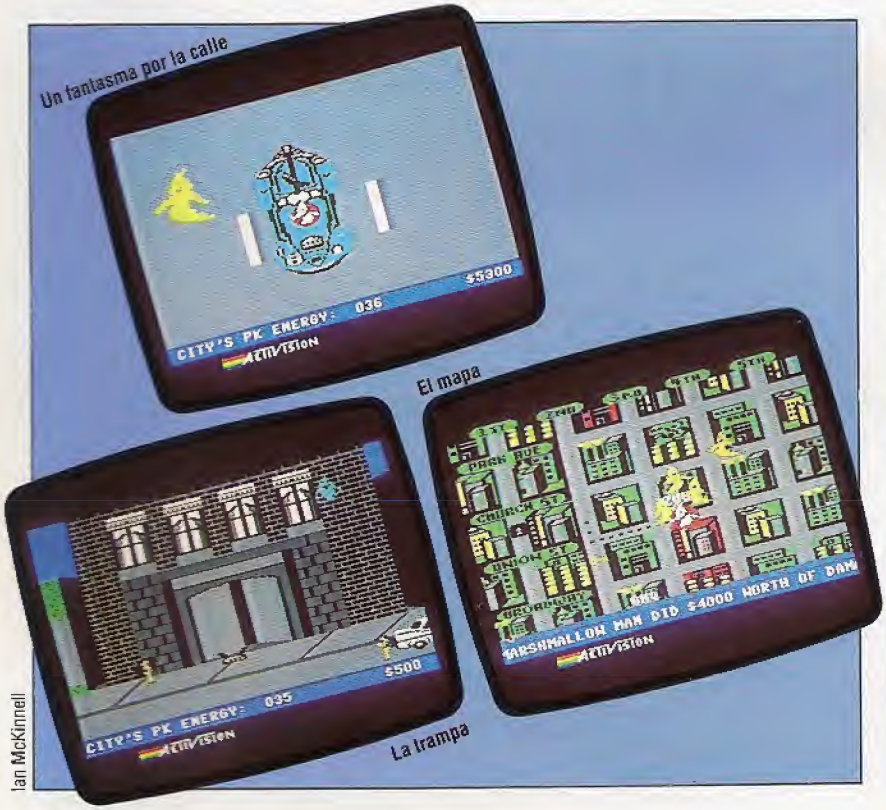
Mientras sucede esto, el jugador debe estar atento a la alerta *marshmallow* (bombón de merengue blando). Esta significa que los *roamers* se están reuniendo para formar al *hombre marshmallow*, quien destruirá la ciudad a menos que se arroje inmediatamente algún cebo para fantasmas.

El jugador también debe vigilar atentamente para detectar si el *gatekeeper* (portero) y el *keymaster* (llavero) están a punto de unir sus fuerzas en el templo de Zuul. Si esto sucediera antes de que el jugador haya conseguido puntos suficientes para introducir los cazafantasmas en el templo, todo estará perdido.

Ghosbusters es un juego bien elaborado, con gráficos finamente detallados y una melodía pegadiza. La salida de habla es un atractivo adicional. El juego ejerce gran atracción y se ha sabido que incluso usuarios hastiados de los juegos por ordenador han pasado horas persiguiendo a los evasivos espectros.

Ghostbusters: Para Commodore 64 y Spectrum
Editado por: Activision (UK) Inc, 15 Harley House, Marylebone Road, Regents Park, London NW1, Gran Bretaña
Autores: David Crane, Adam Bellin, Hilary Mills
Palanca de mando: Necesaria
Formato: Cassette

Fantasmas en la máquina
En la ilustración vemos tres instantáneas del juego *Ghostbusters*, de Activision. Viajando hacia el escenario de una "aparición", se le aparecerán fantasmas por las calles. Usted los puede capturar siempre y cuando haya instalado en el techo de su coche una trampa para fantasmas.
Una vez en el escenario de la aparición, los fantasmas se conducen a una trampa mediante rayos láser. El mapa refleja la situación del templo de Zuul e indica la presencia del *hombre marshmallow*, que deambula por la ciudad destruyendo partes de la misma



Ian McKinnell

Micros en la escuela

Iniciamos una serie en que analizaremos el impacto de los ordenadores en el ámbito de la educación

Tony Lodge

"A largo plazo esperamos que los sistemas de información relacionados con el ordenador tengan tanta influencia en la educación como la que tuvo en su día el desarrollo del libro impreso." Estas palabras, extraídas de un informe del gobierno británico en 1978, las ha repetido recientemente el profesor Tom Stonier, una destacada autoridad en nueva tecnología. Tanto él como otras personas, incluyendo al experto en educación norteamericano Seymour Papert, creen que las escuelas, tal como las conocemos en hoy en día, desaparecerán durante el curso de nuestra vida.

Esta visión más bien radical es una consecuencia natural del nivel de compromiso de estos pedagogos en el campo de la educación informatizada. En la práctica, no obstante, el cambio se está produciendo con gran lentitud y conducido por los educadores y no por las máquinas que tienen a su disposición. Sin embargo, son muchas las personas, tanto pertenecientes al sistema educativo como ajenas al mismo, que piensan que, aplicada adecuadamente, la informática puede causar un impacto significativo en el proceso del aprendizaje.

El empleo del ordenador en la clase se remonta a diversos proyectos y experimentos educativos realizados en el transcurso de los últimos veinte años, si bien algunos de los principios implicados han estado presentes desde mucho tiempo antes. En los años veinte se construyeron máquinas para enseñar simples, si bien no se popularizaron hasta los años cincuenta, cuando B. F. Skinner creó una primitiva máquina de enseñanza por realimentación. Skinner, uno de los psicólogos educativos que más influyeron en los años cincuenta y sesenta, desarrolló sus ideas a lo largo de dos décadas, experimentando con ratas y palomas. Uno de sus proyectos más inusuales involucraba un mecanismo autodireccional para un misil que llevaba tres palomas en su parte delantera, anticipándose de este modo en veinte años al Cruise. Skinner había condicionado a

La versatilidad de una máquina
En la clase de hoy en día, un ordenador puede, con el software adecuado, reemplazar fácilmente la pizarra, el proyector de diapositivas, el televisor, el aparato para cassettes y la biblioteca. Donde no es posible la sustitución directa (el "rincón de pintura" de un aula de primaria, p. ej.) los ordenadores pueden complementar útilmente los métodos de enseñanza tradicionales. Y, lo más importante, el maestro queda liberado de tareas muy caras en cuanto a tiempo (marcar textos, etc.) y dispone de más tiempo para interactuar con cada estudiante sobre una base individual

los pájaros para que picotearan un punto del mapa recompensándolos con alimento. Cabe agregar que el proyecto fue rechazado por el Pentágono, lo que hizo que dejara tras de sí "una enorme cantidad de equipo inútil y unas docenas de palomas con un extraño interés por una zona de la costa de Nueva Jersey".

Las ideas de Skinner acerca del aprendizaje, sin embargo, fueron mejor recibidas. Abogó por un sistema de "programación lineal" (nada que ver con la programación de ordenadores) que implicaba presentarle información a un estudiante en pequeñas dosis cada vez y reforzando positivamente una respuesta correcta. En realidad, Skinner aplicaba a los estudiantes el mismo principio que había ensayado en sus "palomas piloto".

Programación lineal

De la programación lineal surgieron dos conceptos importantes: el *feedback* o realimentación y la *individualización*. Realimentación alude a la respuesta inmediata que se le da al estudiante al decirle si la respuesta dada es correcta o incorrecta. Lamentablemente, las máquinas de aprendizaje de Skinner, cuya operatoria se basaba en manivelas y botones a presión, sólo podía aceptar respuestas específicas. Individualización alude a la naturaleza directa del método, que permite que el estudiante trabaje a su propio paso y no al que le imponen sus compañeros de clase o su maestro.

Los ordenadores se volvieron algo común cuando las máquinas de aprendizaje de Skinner se hallaban en la cima de su popularidad, y pronto se implementaron en los mismos los sistemas de la programación lineal. A pesar de las importantes limitaciones de este método, aún nos encontramos frecuentemente con programas que lo utilizan. "Lo siento, prueba otra vez" o "Bien hecho, lo has conseguido" aparecen centelleando en la pantalla, reforzando las respuestas a un alud de preguntas.

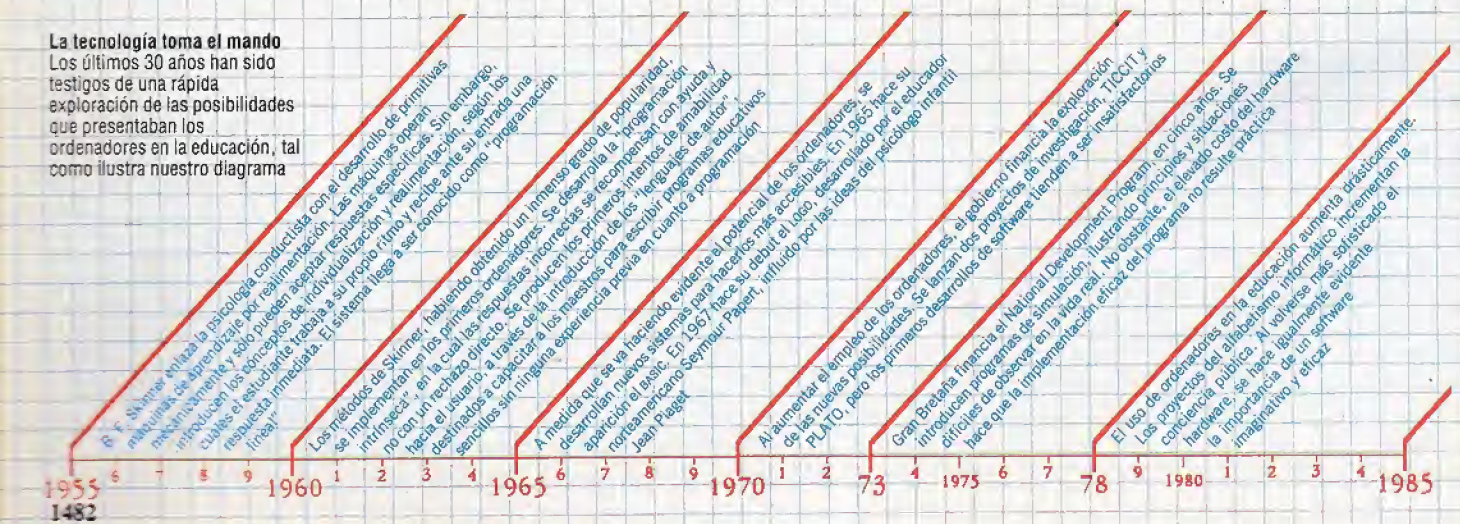
Un desarrollo de la programación lineal fue el *programa ramificado*. En vez de sólo obtener otro intento ante una respuesta incorrecta, el estudiante recibía ayuda y era luego reexaminado. El enfoque parece lógico, pero la "programación intrínseca" (como se la conoció) molestó en gran medida a Skinner y a sus discípulos. Sin embargo, para el estudiante la realimentación era mayor y estaba elaborada de forma más individualizada que antes, y corregía todo malentendido. Los "lenguajes de autor" se idearon para ayudar a los maestros a escribir este tipo de programas. Representaron un temprano intento por llegar a la "amabilidad" con el usuario, capacitando supuestamente a un maestro para escribir un programa educativo sin tener que aprender programación de ordenadores.

Tanto los programas lineales como los intrínsecos poseen graves inconvenientes. Consideran al estudiante como un recipiente vacío que hay que llenar con conocimiento, y ven el aprendizaje como la adquisición de hechos en lugar de experiencias. No dejan lugar a la autoexpresión, la imaginación ni la creatividad, y fomentan una única respuesta "correcta". Estos sistemas, que nos dan el término *aprendizaje programado*, han creado injustamente una mala reputación a todo el aprendizaje asistido por ordenador.

A pesar de la influencia restrictiva de las primeras máquinas para enseñar, se reconoció enseguida que los ordenadores ofrecían posibilidades totalmente nuevas para la educación. Gran parte de los primeros trabajos realizados en este ámbito han tenido una importancia perdurable y no sólo para los maestros de escuela. En 1965, en el Dartmouth College de Estados Unidos se preparó un proyecto para idear un lenguaje de enseñanza "lo más parecido posible al inglés" que les permitiera a los científicos, ingenieros y estudiantes escribir sus propios programas. Debido a las limitaciones de hardware de aquella época, cuando una máquina de 16 K ocupaba toda una sala de estar, hubo de diseñarse el lenguaje de modo que ocupara el menor espacio de memoria posible. El resultado fue el Beginner's All-purpose Symbolic Instruction Code (código simbólico de instrucciones para fines generales destinado al principiante), o BASIC, y ha permanecido con nosotros desde entonces.

Un par de años después de que se creara el BASIC, un equipo de científicos en ordenadores y especialistas de la educación del Massachusetts Institute of Technology, idearon un lenguaje de programación "diseñado para facilitar las cosas al programador en vez de al ordenador". El objetivo fue lograr que los niños pequeños controlaran un ordenador, una idea revolucionaria para aquella época, dado que todos los proyectos previos habían estado dirigidos a estudiantes universitarios. El jefe del proyecto era Seymour Papert, quien había estudiado con el psicólogo infantil Jean Piaget. Las ideas de Piaget acerca de la educación eran radicalmente diferentes de las de Skinner. Contrastando con el

La tecnología toma el mando
Los últimos 30 años han sido testigos de una rápida exploración de las posibilidades que presentaban los ordenadores en la educación, tal como ilustra nuestro diagrama



enfoque mecanicista de este último, Papert creía en la creación de un entorno en el cual el niño pudiera aprender a través del descubrimiento. Propuso que: "En vez de dejar que el ordenador programe al niño, debemos permitir que el niño programe al ordenador." El resultado de su trabajo fue el LOGO, un lenguaje que recientemente ha llegado hasta los micros y que se está popularizando en las escuelas primarias. El advenimiento del BASIC y del LOGO puso la programación de ordenadores al alcance de los estudiantes, incluso de los de nivel primario.

A pesar de los esfuerzos que se estaban dedicando al desarrollo de las técnicas de aprendizaje asistidas por ordenador, muchas personas seguían albergando dudas sobre los beneficios del empleo de la informática en el campo de la educación. En un esfuerzo por despejar parte de esta incertidumbre, la National Science Foundation de Estados Unidos decidió en 1970 invertir 10 millones de dólares a lo largo de cinco años para investigar al tema. Se crearon dos proyectos: TICCIT y PLATO.

TICCIT y PLATO

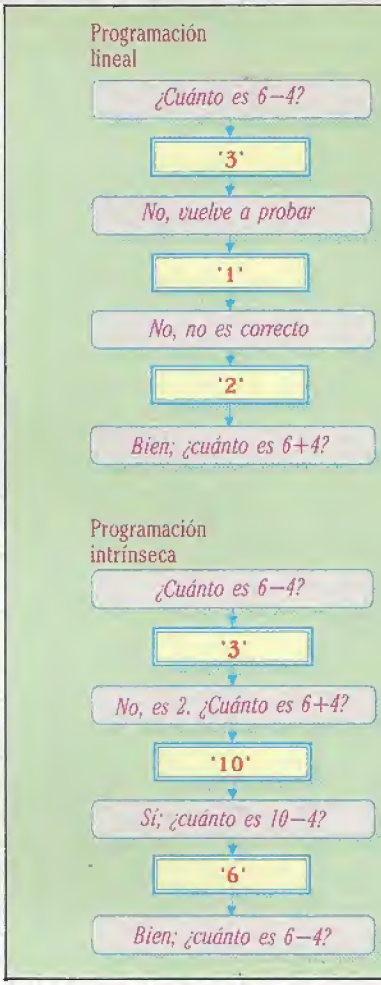
El TICCIT (Time-shared Interactive Computer Controlled Information Television) tuvo como objetivo "proporcionar mejor instrucción a menor costo que la instrucción tradicional en las escuelas de la comunidad". Se le encargó la producción de hardware y software para educación a una empresa, que desarrolló un teclado para estudiantes que incluía mandos especiales de "control para principiantes", una visualización de televisión para texto, gráficos y videos, y un altavoz para mensajes sonoros. Éstos se configuraron en sistemas de 128 terminales, controlados por dos miniordenadores. Hubo problemas con el software y muchos estudiantes no finalizaron el curso, pero quienes lo terminaron obtuvieron mejores calificaciones que aquellos que habían estudiado con métodos convencionales. El TICCIT ha seguido utilizándose en los dos colegios piloto, pero no ha sido adoptado en otro sitio.

PLATO (Programmed Logic for Automatic Teaching Operation) consiste en una red de alrededor de 1 000 terminales conectados a un ordenador central en Illinois. Los terminales se pueden comunicar entre sí y acceder a una biblioteca de lecciones retenida en el ordenador principal. Las unidades de visualización de PLATO poseen paneles transparentes que permiten proyectar diapositivas en color sobre gráficos de ordenador. Un panel al tacto permite a los estudiantes comunicarse con el programa tocando la pantalla. Los maestros pueden escribir sus propios programas en un lenguaje de autor denominado TUTOR. Al igual que el TICCIT, la preparación del software consumió más tiempo y esfuerzo de lo previsto. Los resultados de los estudiantes que utilizaron el PLATO no fueron ni mejores ni peores que los de los estudiantes que emplearon los métodos convencionales, aunque a los que lo aplicaron el sistema les resultó más amable que a los que trabajaron con el TICCIT.

En 1973 el gobierno británico patrocinó un National Development Program para aprendizaje asistido por ordenador, de cinco años de duración y con un presupuesto de 2,5 millones de libras. La mayor parte del dinero se gastó en desarrollar software en forma de simulaciones, utilizándose el ordenador para crear situaciones difíciles o imposibles de observar en la vida real. Un programa, por ejemplo, simulaba la concentración de tintura en el torrente sanguíneo a diversos intervalos de tiempo y otros ilustraban reacciones químicas. No obstante, la mayor parte del software era muy poco imaginativo y el elevado costo de los ordenadores en los años setenta hizo que su uso no fuera práctico.

Todo sistema educativo sufre la influencia de la política, la economía y las demandas de la sociedad. Los ordenadores en la educación no sólo tienen que pugnar contra estos factores, sino también contra las presiones del cambio tecnológico. A ello se debe añadir el conservadurismo de quienes ven a los ordenadores como la simple representación de una extensión de las nada deseables técnicas de aprendizaje programado que analizáramos con anterioridad.

Mientras tanto, al igual que en todas las áreas de la informática, las tendencias dominantes de hoy en día en la informática educacional son consecuencia de la constante caída del precio de la memoria. Hoy en una escuela primaria de Londres hay más potencia informática que en todas las universidades británicas de hace 20 años. Un efecto de ello, según algunos críticos, ha sido la concentración de adquisición de hardware al costo del desarrollo del software. Sin embargo, ahora los políticos parecen más preparados para reconocer el software como un recurso nacional de importancia vital y, en la medida en que el precio del hardware continúe disminuyendo, podemos esperar ver cambios radicales en los sistemas educativos en los próximos diez años.



Una mano amiga

Utilizando los sistemas de programación lineal, el estudiante debe emitir la respuesta correcta a una pregunta antes de poder seguir adelante. El sistema rechaza las respuestas incorrectas y luego vuelve a presentar la pregunta original sin ofrecerle al estudiante ninguna otra ayuda. Con los sistemas de programación intrínseca, sin embargo, una entrada equivocada por parte del estudiante no conducirá simplemente a que se vuelva a plantear de inmediato el mismo problema. En cambio, el sistema responderá con un mensaje de error y una serie de preguntas diferentes, aunque relacionadas con la anterior. Después de que el estudiante responda correctamente a éstas, se le vuelve a plantear el problema original.

Primeras palabras

En este capítulo de nuestra serie sobre el lenguaje PASCAL examinaremos algunos de los fundamentos de su sintaxis y su vocabulario

Compilación popular
Una implementación profesional completa del PASCAL puede alcanzar un precio que supere al de un ordenador personal, pero recientemente se ha producido una ola de compiladores de precio razonable para micros personales populares. Mostramos una selección de los paquetes disponibles en la actualidad

El problema que por norma general se plantea primero cuando se utiliza un lenguaje compilado es aprender a abordar el proceso de múltiples etapas requerido para obtener siquiera un pequeño programa que sea ejecutable. Primero se ha de entrar el texto fuente a través de algún tipo de editor de texto o procesador de texto. Luego, tras haber guardado el fuente en cinta o disco, se debe cargar el compilador del lenguaje e instruirlo para que compile el fuente a alguna forma de código máquina (a menudo con una compleja cadena de opciones de líneas de comando). Por último, este archivo "objeto" se debe "relocatar" y enlazar con las rutinas de biblioteca en *run-time* necesarias. Posiblemente el programa se pueda cargar y ejecutar sin ningún otro esfuerzo, pero si el compilador genera un "seudocódigo" o código intermedio, entonces para ejecutar el programa se habrá de utilizar un intérprete en *run-time*.

Si todo esto le suena excesivamente complicado, le tranquilizará el hecho de que todos los paquetes de PASCAL disponibles para ordenadores personales evitan en gran medida estos problemas. Al menos durante el desarrollo del programa, su texto fuente, compilador y programa objeto pueden estar residentes en la memoria. Ello es posible gracias a la eficacia y el pequeño tamaño del PASCAL, y el sistema resultante con frecuencia no es más complicado, al utilizarlo, que un paquete de BASIC residente. Sólo cuando comencemos a escribir programas de tamaño significativo necesitaremos recurrir a los procesos más complejos que hemos descrito.

Cada sistema posee su propio juego de instrucciones para controlar el editor y el compilador, y usted debe remitirse a la documentación correspondiente. Con mucha frecuencia, todo cuanto se requiere es una simple E para editar, C para compilar y R para ejecutar. Lo que nos interesa es la sintaxis correcta para entrar un programa, independientemente de lo trivial o complejo que el mismo pueda ser. Por suerte, el PASCAL está tan estandarizado que casi no serán necesarios los muchos "complementos" a la programación que hemos venido ofreciendo para las distintas versiones de BASIC (si bien más avanzado el curso sí tendremos que incluir algunas excepciones menores). Así pues, veamos nuestro primer programa completo en PASCAL:

```
Program Uno (output);
Const
    Mensaje='Programacion en Pascal';
Begin
  write (Mensaje)
End.
```

Antes de estudiar en profundidad este ejemplo, pruebe de entrarlo, compilarlo y ejecutarlo. Si obtiene alguna queja por parte del compilador, lea atentamente el mensaje de error y vea si puede detectar qué es lo que está mal. Todos los símbolos se deben presentar exactamente tal como se indican. Tenga especial cuidado con el punto y coma tras la primera y la tercera líneas, y el punto al final del programa. Cuando haya ejecutado con todo éxito este programa, verá visualizado en la pantalla el siguiente mensaje:

```
Programacion en Pascal
```

El programa es trivial, pero demuestra la forma general que tendrá cada módulo (programa, procedimiento o función). Hay tres partes separadas:
1. El encabezamiento, en este caso un encabezamiento de programa.
2. Declaraciones y definiciones, aquí hay una única definición de constante.
3. El "cuerpo", que contiene todas las sentencias ejecutables.

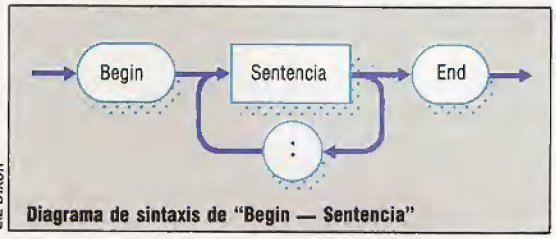
Las exigencias de sintaxis del PASCAL, al menos para los fundamentos del lenguaje, se pueden definir mejor mediante "diagramas de sintaxis". Éstos son como el mapa de carreteras de un sistema de una sola dirección. La ruta legal a través del diagrama avanza desde arriba a la izquierda hasta abajo a la

Paquete: HISOFT PASCAL
Para: Spectrum, Amstrad, máquinas MSX
Descripción: Una implementación ligeramente no normalizada, pero, no obstante, una ganga. El paquete viene con su propio editor y con una biblioteca de gráficos de tortuga

Paquete: Acorn ISO PASCAL
Para: BBC Modelo B, Electron
Descripción: Una oferta excelente. Hay dos compiladores: uno en ROM, con un editor semiinteligente y compilación de memoria a memoria; el otro es un sistema basado en disco para quienes posean el segundo procesador (6502)

Paquete: Oxford PASCAL
Para: Commodore 64
Descripción: Otra ganga. Se espera que Limbic Systems, la empresa que produce el paquete, saque al mercado versiones para el BBC Micro y el Spectrum en fechas muy próximas

derecha, y todos los casilleros que atravesemos son o bien una "entidad sintáctica" (es decir, que se representa a sí misma) contenida en un casillero de bordes redondos, o bien otro elemento descrito en alguna otra parte mediante un diagrama de sintaxis diferente, indicado por el casillero rectangular que lo contiene.



Liz Dixon

Diagrama de sintaxis de "Begin — Sentencia"

Refiriéndonos primero al diagrama global de un programa, se puede apreciar que las palabras Begin y End están definidas como integrantes del vocabulario del PASCAL, y para elaborar su significado no se requiere ningún otro diagrama. De hecho, el PASCAL reconoce sólo 35 palabras con un significado fijo, cuya lista completa ofrecemos aquí por razones de completitud. Nuestro primer programa sólo hace uso de cuatro de ellas: Program, Const (de *constant*: constante), Begin y End. La palabra que sigue a Program es un "identificador" que identifica el nombre del programa y puede ser cualquier identificador legal que se elija.

Comenzando por una letra y usando sólo letras o dígitos, la cantidad de nombres que podríamos utilizar es inmensa. Sin embargo, como es lógico, es imposible emplear una palabra reservada. P. ej.:

```
Nombre
PASCAL
ProgramUno
N
XYZ123
Direccion12
FloraMacDonald
UnIdentificadorMuyLargoRealmente
```

son todos legales, mientras que:

```
Prog-1
DIEZ%
Trastos.Jack
and
Time$Square
No-lo-se
101Dalmatas
Es igual a
```

son todos ilegales, bien porque poseen símbolos que no son alfanuméricos, bien porque empiezan por un número o (en el caso de and) son alguna de las palabras reservadas del PASCAL. El último ejemplo es ilegal porque se utiliza un espacio para separar palabras, siendo legales las palabras que lo componen (Es, igual y a). En PASCAL, como en castellano o inglés, las letras en minúscula y en mayúscula no suponen ninguna diferencia de significado, si bien algunas versiones no estandarizadas exigen que las palabras reservadas vayan en mayúsculas.

Aparte de los espacios y el final de una línea, en la sintaxis del PASCAL hay otro elemento que se puede emplear como separador: un comentario. Éste puede aparecer en cualquier lugar del texto, excepto, por supuesto, en medio de las palabras. Los comentarios se delimitan mediante "llaves" ({}).

Seamos más audaces y entremos algo que se parezca más a un auténtico programa en PASCAL:

```
Program ProgramDos (input,output);
        {Da el cuadrado de un numero}
Const
        Aviso='Entre un numero:';
Var
                numero :integer;

Begin
   WriteLn;
   WriteLn;
   write (Aviso);
   read (numero);
   WriteLn (numero,'al cuadrado es',numero*numero)
End.
```

Observe que ahora incluimos en el encabezamiento el identificador input. El PASCAL requiere input y output, que identifican los archivos externos con los cuales se comunicará el programa. En un micro, normalmente serán el teclado y la pantalla, respectivamente. Mediante la inclusión de input podemos ahora leer desde el teclado gracias al procedimiento estándar read. Al igual que write, todos los "parámetros" se deben listar entre paréntesis.

La memoria utilizada para almacenar estos parámetros queda reservada por la declaración Var, en este caso para un único número entero. A diferencia del BASIC, que por lo general sólo puede distinguir entre números y series (mediante el empleo de un signo dólar después del identificador), la gama de tipos de datos de que dispone el PASCAL es casi ilimitada. Por consiguiente, es importante informar al compilador sobre cuánta memoria debe reservar para almacenar cada dato. Usted siempre debe declarar las variables simples en una declaración Var.

El cursor permanecerá posicionado inmediatamente después del aviso, igual que si hubiéramos empleado una sentencia PRINT del BASIC con un punto y coma final. Esto es exactamente lo que queríamos al utilizar el procesamiento incorporado write del PASCAL. Siempre que se requiera una nueva línea, debemos emplear el procedimiento alternativo, WriteLn. La Ln es una contracción de la palabra Line (línea), y es útil usar una W y L mayúsculas para indicar el comienzo de cada una de las palabras que lo componen. Una sentencia WriteLn simple sin parámetros creará sólo una nueva línea.

El PASCAL nos permite diferenciar entre datos que varían y datos que permanecen constantes a lo largo de la ejecución del programa. Observe que la definición Const emplea un signo de igualdad, mientras que la declaración Var utiliza dos puntos.

Las 35 palabras del PASCAL

And	Case	Do	End	Function	In	Not	Procedure	Set	Until
Array	Const	Downto	File	Goto	Label	Of	Program	Then	Var
Begin	Div	Else	For	If	Mod	Or	Record	To	While
					Nil	Packed	Repeat	Type	With

Hacia adelante

He aquí otro paquete de análisis del camino crítico, esta vez sobre máquinas como el IBM-PC y compatibles

Aunque el paquete *Project* de Microsoft no posee la clase de facilidades para gráficos de *MacProject* (o de su programa rival, *Graphics environment manager*, de Digital Research), hace el mejor uso posible del limitado potencial gráfico del sistema IBM. Por ejemplo, las combinaciones de signos de igualdad o de guiones con el signo de mayor puede producir algunas útiles flechas:

==========> ----------

La primera se puede visualizar en negrita para indicar los caminos críticos, y la segunda en un tipo más suave para reflejar aquellas actividades que no son críticas. Sin embargo, los factores más sobresalientes que se deben considerar son de una naturaleza más comprometida que ésta. Ellos son:

- ¿Cuál es el grado de flexibilidad del programa, y cuál es su capacidad de respuesta a la clase de cambios que probablemente sean necesarios a medida que se desarrolle el proyecto?
- ¿Con qué claridad visualiza las relaciones existentes entre las actividades, y en qué medida contribuye la misma a la eficaz dirección del proyecto?
- ¿Se puede utilizar el programa como medio auxiliar para la planificación inicial, o los usuarios deben primero desarrollar por completo sus ideas?

Respondiendo primero a la última pregunta, el usuario en realidad necesita clasificar el proyecto sobre el papel antes de digitar la primera actividad. De hecho, el manual afirma de forma inequívoca:

"Antes de entrar actividades en el Microsoft Project, ha de disponer de la siguiente información sobre cada actividad:
En qué consiste (descripción)
Cuánto lleva (duración)
Qué se ha de hacer con anterioridad (predecesores)
Cuándo comienza (fecha de inicio)."

Para poner de relieve la importancia de tal preparación, el manual ofrece un apéndice de diez páginas, *"Defining tasks by a work breakdown structure"* (Definiendo las tareas mediante una estructura de análisis del trabajo), que afirma:

"La dirección exitosa de un proyecto empieza al comienzo de un proyecto. Primero, debe enunciar claramente los objetivos, la finalidad, las limitaciones y las especificaciones del proyecto. Luego debe definir el conjunto de actividades específicas que constituyen el proyecto.

"Ningún programa para ordenador puede compensar las actividades definidas de forma inadecuada o sin precisión. La definición de las actividades puede tener implicaciones en la planificación de los recursos, su asignación y su control, así como en el control de costos y presupuesto. La definición inexacta de las actividades probablemente ocasione graves problemas una vez iniciado el proyecto."

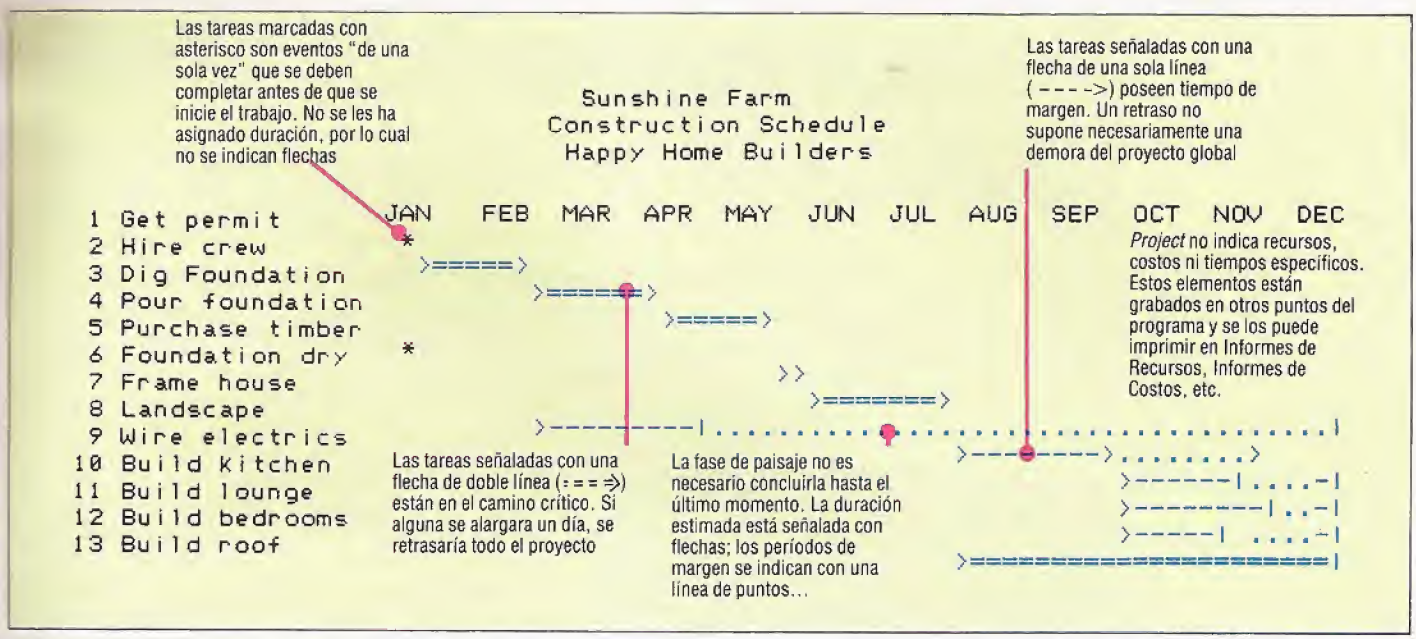
Es difícil que quienes posean experiencia en técnicas de dirección se muestren en desacuerdo con la necesidad de una preparación adecuada. Pero quienes ven al ordenador como una herramienta cualitativamente diferente de todas las otras, actuando como una extensión de la mente del usuario y no como una extensión de las yemas de sus dedos, quizá piensen que la tarea del software es, precisamente, contribuir a evitar actividades definidas inadecuada o inexactamente. Pero, por cierto, Microsoft Project ayuda al usuario a detectar las prioridades. Incluyendo el recurso inicial del lápiz y el papel, también ofrece al usuario muchísimas oportunidades para introducir cambios.

Cuando se conecta por primera vez el ordenador, la fecha y la hora, que se incluyen como parte de secuencia de entrada *(sign-on)*, determinarán el calendario empleado por el programa; pero el calendario se puede cambiar con bastante facilidad, incluso tras haber digitado varias actividades. De alterarse la hora asignada a alguna actividad, tan pronto se entra la modificación, se vuelve a calcular el diagrama del camino, cambiando algunas flechas de actividades a negritas ===> (críticas), otras a ligeras -----> (no críticas). Ello no sólo hace que observar la relación entre actividades sea muy fácil, sino que también destaca de forma instantánea un cambio que, de lo contrario, no le habría resultado obvio al usuario. Por todo ello, parece que Microsoft ha aplicado la experiencia que obtuvo con la producción del *Multiplan*, una de las hojas electrónicas de mayor éxito, después de que en 1979 el pionero *Visicalc* inaugurara el género. En realidad, usar el *Microsoft Project* es casi como utilizar una hoja electrónica, y quienes estén familiarizados con el empleo de las mismas se sentirán casi de inmediato a sus anchas con *Project*.

Tras una breve visualización del logotipo de Microsoft, se visualiza la pantalla de actividades (vacía, si no se hubiera especificado ningún nombre de archivo), con números del 1 al 19 a la izquierda de cada fila y la escala de tiempo a lo largo de la parte superior, empezando por la fecha de hoy, si se ha entrado inicialmente. Ésta se puede modificar mediante la entrada OPTIONS del menú de la parte inferior de la pantalla (nuevamente al estilo de *Multiplan*), que incluye asimismo:

BLANK CALENDAR DELETE EDIT GOTO HELP INSERT MOVE PRINT QUIT RESOURCE SORT TRANSFER XTERNAL

Debajo de la frase *"Select option or type command letter"* (Seleccione opción y pulse letra de comando), se visualiza la opción en uso en ese momento (inicialmente ACTIVITY) y, en el extremo inferior derecho, el nombre del *Project* en uso, al cual de forma automática se le da el título TEMP hasta que el usuario le asigne algún otro.



Proyecto de construcción
Este diagrama representa un plan para la construcción de una casa. Las fechas aparecen a través de la parte superior del diagrama, y cada una de las tareas aparece, por orden de operación, a lo largo del margen izquierdo. La duración de cada tarea se representa mediante flechas construidas con caracteres del teclado (<>.=-)

La escala de tiempo se puede regular en días, semanas o meses y se la puede cambiar en cualquier momento para contribuir a ofrecer una visión general de todo el proyecto a simple vista. Con una escala de tiempo diaria sólo se pueden ver, supongamos, las fechas entre el 1 de enero y 2 de marzo, mientras que la escala de tiempo mensual permite ver a simple vista dos años y tres meses. Las teclas derecha e izquierda para control del cursor se emplean para desplazar hacia los lados la pantalla ACTIVITY, al estilo habitual de una hoja electrónica. Mediante la opción CALENDAR, el usuario programa primero las vacaciones y otros días no laborables; los sábados y domingos normalmente se planifican como días festivos. Ello, sin embargo, se puede alterar, pero se debe realizar para cada día individual. Los meses posteriores se pueden visualizar mediante la tecla <PAGE DOWN>, y los meses anteriores se pueden recuperar con <PAGE UP>.

Al igual que sucede con la mayoría de las opciones de fecha en el IBM y máquinas compatibles, debe tenerse cuidado en utilizar el orden norteamericano de MM/DD/AA (p. ej., 01/07/85 es 7 de enero de 1985, y no 1 de julio de 1985). Una vez ajustado el calendario, puede ser guardado (SAVE) utilizando la opción TRANSFER en el estilo *Multiplan* habitual. El programa añade el sufijo .CAL para distinguirlo de los archivos de actividades (sufijo .ACT) y de recursos (.RES).

El usuario comienza luego a digitar actividades, el tiempo que lleva cada una, y anunciando las actividades previas de las que depende (sus "predecesoras"), con las dependencias múltiples separadas mediante comas. La columna de Activities Titles (nombres de las actividades) está limitada a 15 caracteres y no se puede ensanchar. Sin embargo, de digitarse más de 15 caracteres, éstos se retendrán en la memoria y se incluirán en las salidas impresas. A medida que se va digitando cada actividad (con su duración, su(s) predecesora(s) y su fecha de inicio), se van visualizando inmediatamente las relaciones entre las diversas actividades, con el tiempo de margen (período durante el cual se puede retrasar una actividad sin que se atrase el proyecto total) representado en forma de puntos. En este momento se asignan, asimismo, los recursos. Apenas se entra un recurso, se puede llamar a la Resource Table (tabla de recursos) para controlar los resultados.

Una vez entrados los datos, se puede imprimir el plan completo en una base de página a página. Pero si se ha instalado una ficha para gráficos y conectado la impresora apropiada (impresora para gráficos del IBM-PC o las Epson MX-80, MX-100, FX-80 o FX-100), se puede volcar todo el proyecto de forma apaisada y obtener una impresión a lo largo de la longitud del papel. Esto es útil, dado que significa que se puede imprimir un plan completo sin que sea necesario cortar y pegar las salidas impresas.

Asimismo, es posible obtener informes detallados acerca de cada una de las actividades, reflejando las fechas de inicio y terminación más tempranas y más tardías, los recursos, las predecesoras y los tiempos de margen, y una tabla de actividades para el proyecto en conjunto visualizando la misma información pero de forma menos detallada.

En la práctica, la carencia del *Microsoft Project* de gráficos sofisticados representa una desventaja menos importante de lo que se podría pensar en un primer momento, y aunque el manual (excelente) hace hincapié en la necesidad de una preplanificación adecuada, aun así se puede ajustar la información después del segundo o tercer cambio de idea o tras demoras inesperadas. El programa será particularmente amable para quien tenga experiencia con hojas electrónicas, en especial el *Multiplan*, porque su método de operación es muy similar. En operación el *Microsoft Project* es más rápido que el *MacProject* de Macintosh, pero su precio es casi tres veces superior.

Microsoft Project: Para máquinas MS-DOS
Distribuido por: Microsoft Ltd, Piper House, Hatch Lane, Windsor, Berkshire, Gran Bretaña
Autores: MAS, Seattle, Estados Unidos
Formato: Disco

Se reanudan los juicios

Nuestra comentarista Stephanie Brittain hace una valoración crítica de dos recientes programas de juegos escritos para el Commodore 64

Percy the potty pigeon

(Percy, la paloma chiflada)
Gremlin Graphics

Conducir en Londres entraña sus peligros, incluyendo los traumas que puede causar la conocida aversión de las palomas a volar por donde muy bien podrían ir andando.

Percy, la paloma de este juego, está completamente chiflada. Para ella no hay nada mejor en el mundo que volar por sobre la línea del horizonte de la gran ciudad arrojando bombas a los coches que circulan por la autopista: una sensata narración de carnicería vial. Si se la controla mediante la palanca de mando, Percy puede que no sea destructiva del todo: le agrada abalanzarse sobre la autopista (evitando a los vengativos motoristas) para recoger ramitas, que luego lleva hasta el nido más cercano de los árboles que hay junto al borde del camino. En los niveles de juego superiores, la mano que maneja la palanca de mando debe evitar a los gorriones que roban las ramitas y a los halcones que roban la paloma.

Ciertamente, *Percy the potty pigeon* no es un programa por el que Gremlin Graphics pueda sentirse demasiado satisfecha.

Ian McKinnell

Caesar the cat

(El gato Caesar)
Mirrorsoft

Caesar es un pequeño y astuto gatito negriblanco cuyo objetivo consiste en limpiar de ratones la "despensa".

La despensa en cuestión en realidad es una pantalla repleta de alimentos y cacharros por la cual campan a sus anchas los ratones, comiéndose la comida hasta que Caesar los atrapa.

Se pierden puntos si Caesar se estrella contra una "pared" lateral de la pantalla, choca contra algún cacharro o tarda demasiado tiempo en capturar su presa.

Una vez que consigas dominar el primer nivel, puedes perseguir ratones más veloces en los dos niveles superiores: y es aquí donde comienza realmente la diversión. Ciertamente, Mirrorsoft ha acertado de lleno con esta aventura de felino y roedores. No existe ningún juego comparable a éste en cuanto a originalidad. ¡Les gustará hasta a los amantes de los perros!

En resumen, *Caesar the cat* constituye un auténtico acierto por parte de los programadores de Mirrorsoft.

Portátil avanzado

ACT ha utilizado la más reciente tecnología para darle al Apricot Portable un aire futurista

La máquina portátil representa una de las áreas de mayor crecimiento de la microinformática. Se pretende que el hombre de negocios pueda desplazarse con su micro por el mundo y trabajar allí donde haya una adecuada fuente de alimentación eléctrica. En la práctica, no siempre es así como se utiliza la máquina. A menudo, el ordenador portátil se conserva en casa y el trabajo realizado con una máquina compatible en la oficina se lleva a casa por la tarde para acabarlo.

ACT (Apricot) es uno de los más recientes entre una larga serie de fabricantes de ordenadores que ha producido una máquina portátil que es compatible con su muy vendida línea de ordenadores de sobremesa.

El Apricot Portable se suministra en una sólida carcasa plástica cuyo tamaño y forma son el de un maletín, pero con un saliente en el fondo para dar cabida a la pantalla. La carcasa se abre presionando dos cierres a los lados del asa, aunque en la máquina que examinamos ello resultó bastante dificultoso.

Análisis del hardware

En el interior, el teclado y el ordenador se sostienen a cada lado de la carcasa mediante tiras. Con la caja también se proporcionan dos manuales, un micrófono y un ratón/bola de mando. Lo curioso es que no se suministra ningún cable para unir el teclado y el ratón al ordenador. Ello se debe a que los datos se transmiten a través de un haz infrarrojo. La información se transmite y se recibe mediante dos pequeños focos que hay en el borde trasero del teclado y en la parte delantera del ratón. Las señales emitidas por estos dispositivos las recoge un dispositivo receptor que hay en el ordenador. De este modo, el usuario puede trabajar con el teclado sobre el regazo y tener la máquina sobre una mesa a algunos metros de distancia.

Este sistema infrarrojo posee dos desventajas. En primer lugar, si en una misma habitación se estuviera operando con varios Apricot Portable, las señales infrarrojas de las diferentes máquinas podrían interferirse mutuamente. Para evitar esto, Apricot ha proporcionado un tubo ligero. Éste es un cable de fibra óptica que conecta el teclado con el ordenador de la forma convencional. El segundo problema aparece cuando se utiliza el ratón. Si el usuario coloca el teclado directamente frente al ordenador, con frecuencia bloqueará las señales provenientes del ratón. Ello se debe a que el foco receptor de la máquina queda casi al ras de la superficie de la mesa. Por otra parte, si corre el teclado hacia un lado, puede que el ordenador no sea capaz de captar sus señales.

Uno de los problemas de producir una versión portátil de una máquina de sobremesa es comprimir todas las funciones en un espacio más reducido. El teclado plantea dificultades especiales y los fabricantes se han valido de toda clase de técnicas ingeniosas para empaquetar todas las funciones necesarias en un teclado más pequeño y más ligero. Al traducir el teclado Apricot a la versión portátil, el fabricante ha prescindido de la visualización de calculadora LCD y ha colocado las 87 teclas una bien junto a la otra. El trazado es idéntico al del Apricot de sobremesa, a excepción de las teclas de función, que están situadas en el extremo derecho en vez de arriba del trazado principal.

En parte para compensar la pérdida de la visualización LCD, que en la versión de sobremesa también se utiliza para intrucciones que requieren una única pulsación de tecla, Apricot ha añadido una tecla de función extra, lo que hace un total de nueve, y una tecla para visualizar la hora y la fecha. Las teclas QWERTY del lado izquierdo del teclado están flanqueadas por teclas de control, estando las teclas del cursor abajo a la derecha. Entre las teclas de máquina de escribir y las teclas de función hay un teclado de calculadora, con las teclas numéricas en una tonalidad de gris más clara que la del resto del teclado.

Si bien el aspecto del teclado es sumamente elegante, quizá las teclas estén situadas demasiado cerca las uñas de las otras como para resultar cómodas. En vez de las convencionales teclas indentadas, el Portable posee un teclado nivelado estilo QL. Ello hace que el teclado sea más compacto, dado que las teclas no sobresalen del cuerpo principal, pero a muchos mecanógrafos usar este diseño

Chris Stevens

Tecnología de vanguardia
El Apricot Portable utiliza la tecnología más reciente. La máquina está equipada con una pantalla para textos de 80 por 25, que permite visualizar una amplia variedad de aplicaciones MS-DOS. En la fotografía también vemos el teclado y el ratón/mando de bola: éstos no están conectados al ordenador en el sentido convencional, sino que transmiten señales a través de un haz infrarrojo. En el lado derecho está el micrófono, que le proporciona a la máquina la facilidad de reconocer la voz

de teclado les resultará difícil, puesto que a los dedos les es difícil diferenciar las teclas. Lo compacto de esta disposición pone de relieve este problema: no hay ningún vacío entre las teclas de máquina de escribir, las de calculadora y las de función. No obstante, el trazado posee sus compensaciones. En razón del diseño, Shift, Stop y Caps Lock son el doble de grandes que las del Apricot, lo que las hace más fáciles de encontrar.

Encima del teclado y en el lugar de la visualización de calculadora hay cuatro botones para llevar a cabo funciones extras. Hay un botón de Reset, que carga de nuevo el disco que esté en ese momento en la unidad de disco. El segundo botón le permite al usuario modificar la velocidad de repetición de las teclas. El tercero llama a una rutina que pone a cero la hora retenida en el ordenador, y el cuarto traba el teclado. Este último es útil cuando se está utilizando el ratón o la entrada por habla y no desea que la pulsación accidental de una tecla produzca la entrada de información falsa.

La pantalla LCD incorporada está instalada en el cuerpo principal del ordenador. Su nitidez es más pobre que la de máquinas comparables, aun cuando el claro fondo verde oliva sea más brillante que la mayoría. A la derecha de la pantalla hay un micrófono, que se conecta mediante un delgado cable que se enchufa en un conector microjack situado a la izquierda del ordenador.

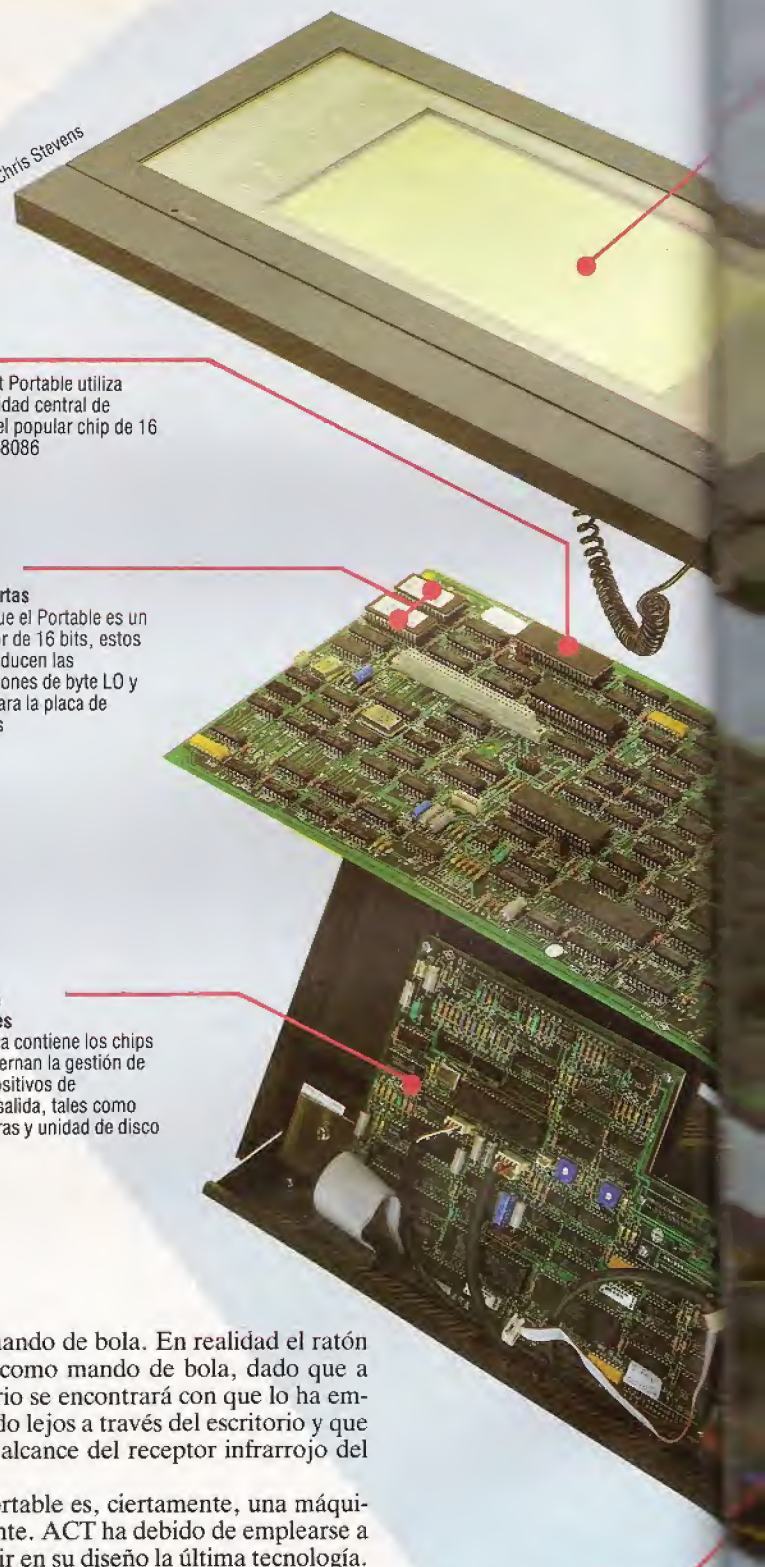
Reconocimiento de voz

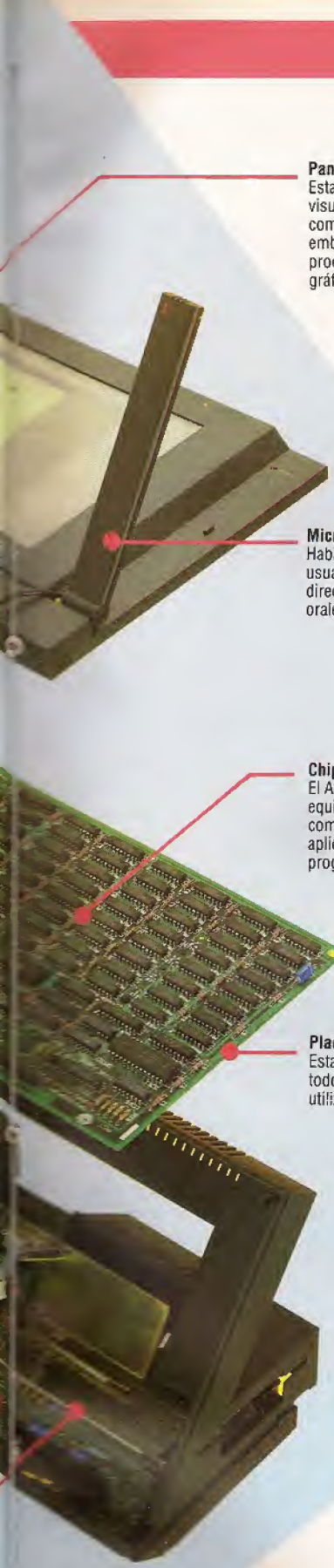
Quizá la característica más interesante del Apricot Portable sea su programa para reconocimiento de voz, que reconoce instrucciones orales, las pasa a la entrada y después las ejecuta de la forma normal. La modalidad Voice Driven Application (aplicación activada mediante la voz) acepta vocabularios de hasta 63 palabras para cualquier programa de aplicaciones, como el *WordStar* o el *SuperCalc*. Un programa de entrenamiento de voz le preguntará al usuario si la voz a utilizar será masculina, femenina o infantil, y si habrá o no mucho ruido de fondo. Después de pronunciar varias veces una instrucción, el ordenador aceptará y comparará diversas variaciones. Sin embargo, aun así malinterpretará palabras y con frecuencia también las ignorará. Por este motivo se recomienda no emplear de forma oral DELETE y FORMAT.

Al igual que el resto de la gama Apricot, el Portable utiliza discos Sony de 3 pulgadas para retener los programas de aplicaciones. La única unidad de disco está situada en el lado derecho del ordenador. En la parte posterior de la máquina hay tres interfaces cubiertas por una carcasa plástica protectora. Éstas son una ranura en paralelo Centronics para impresoras, un conector RS232 para un modem externo u otro dispositivo en serie y una puerta para palanca de mando tipo Atari. Esta ranura para palanca de mando no se provee para juegos sino para instalar un ratón o un mando de bola en aquellas circunstancias en las que el ratón infrarrojo no resulte adecuado.

El ratón infrarrojo activado a pilas que se suministra con el Portable está diseñado de forma inteligente, de modo que pueda ser utilizado bien como un ratón que se pueda deslizar por la superficie de una mesa o escritorio, o bien, invertido, para sostenerlo en la mano o mantenerlo en una posición fija y usarlo como mando de bola. En realidad el ratón funciona mejor como mando de bola, dado que a menudo el usuario se encontrará con que lo ha empujado demasiado lejos a través del escritorio y que se ha salido del alcance del receptor infrarrojo del ordenador.

El Apricot Portable es, ciertamente, una máquina muy interesante. ACT ha debido de emplearse a fondo para incluir en su diseño la última tecnología. No obstante, a muchos usuarios de gestión esta tecnología de vanguardia no les despertará un excesivo interés; lo que ellos desean es una máquina fiable que ejecute sus programas de aplicaciones fáciles y cómodamente. En este sentido el Apricot Portable no se destaca de manera especial. La dificultad de leer la pantalla y la fiabilidad de los dispositivos de entrada podrían hacer que los usuarios de gestión se decantaran hacia una tecnología ya probada y comprobada.



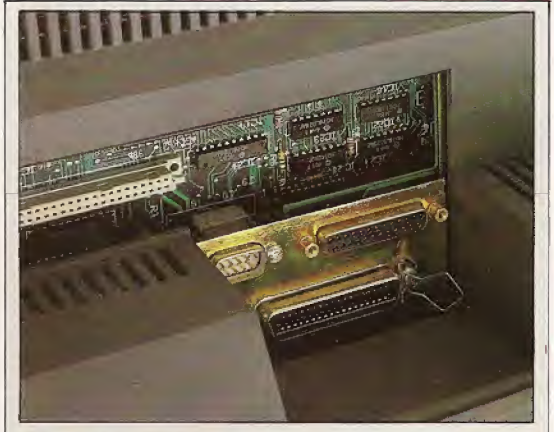


Pantalla LCD
Esta pantalla ofrece una visualización de textos completa de 80 por 25. Sin embargo, también es posible producir en la pantalla gráficos de alta resolución

Micrófono
Hablando por el micrófono, el usuario puede dar directamente instrucciones orales al ordenador

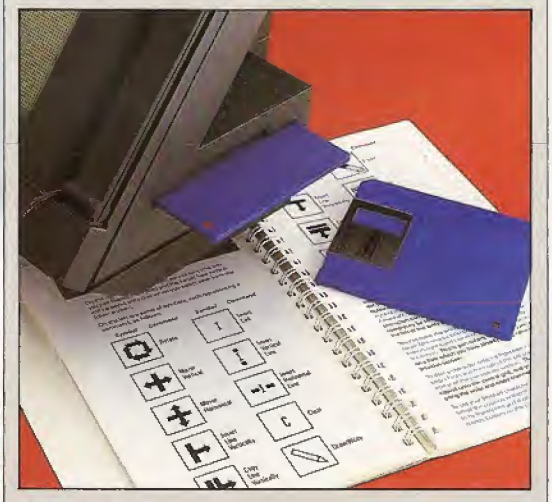
Chips de RAM
El Apricot Portable está equipado con 256 K completos de RAM para aplicaciones de gestión y programas en BASIC

Placa CPU
Esta placa contiene la CPU y todos los otros chips que utiliza el procesador



Interfaces para periféricos
En la parte posterior del ordenador, protegidas bajo una carcasa plástica, están las interfaces para periféricos. Abajo está la puerta en paralelo Centronics y, encima de ella, a la derecha, el conector RS232 (que permite conectar un modem externo). A la izquierda hay una puerta a través de la cual se puede conectar un ratón o un mando de bola convencionales

Mando de bola infrarrojo
Haciendo girar la bola, la flecha del cursor se desplaza en la dirección correspondiente a través de la pantalla. Cuando el cursor está señalando el icono elegido, pulsando uno de los botones del costado se accede a la aplicación escogida. El dispositivo se puede dar vuelta, de modo que la bola ruede, lo que le permite al dispositivo actuar a modo de ratón



Chris Stevens

Software Apricot
Con el ordenador se suministran algunos paquetes de aplicaciones que permiten ejecutar varios programas diferentes. Estas aplicaciones están retenidas en discos Sony de 3 pulgadas. Entre los paquetes están el *SuperCalc* (hoja electrónica), el *SuperPlanner* (fichero de tarjetas y calendario) y el *SuperWriter* (procesador de textos)

APRICOT PORTABLE

DIMENSIONES
450 x 335 x 200 mm

CPU
Intel 8086

MEMORIA
32 K de ROM y 256 K de RAM, de los cuales hay 211 K libres para programas de aplicaciones

PANTALLA
Pantalla LCD electrofluorescente con una resolución de texto de 80×25 caracteres y una resolución para gráficos de 640×256 pixels. Si se enchufa el ordenador a una pantalla externa, se pueden visualizar hasta 16 colores

INTERFACES
En serie RS232, en paralelo Centronics y una conexión para ratón. También hay una conexión de "tubo de luz" para utilizar el teclado basado en infrarrojos y un conector hembra para el micrófono

UNIDADES DE DISCO
El Portable contiene una unidad para discos Sony de 3 pulgadas, doble densidad y 720 K

SISTEMAS OPERATIVOS
MS-DOS, CP/M-86 y Concurrent CP/M

TECLADO
Teclado compatible con IBM, conteniendo 87 teclas con diez teclas de función. Hay, asimismo, cuatro botones de función

DOCUMENTACION
Un manual Starter Pack, que ofrece detalles sobre cómo instalar el ordenador y cómo utilizar el sistema operativo y las facilidades para reconocimiento de voz. El manual Applications Pack ofrece una guía para las aplicaciones de *SuperCalc*, *SuperPlanner* y *SuperWriter*

VENTAJAS
El Apricot Portable es una de las máquinas más avanzadas que existen en la actualidad

DESVENTAJAS
Debido a que la tecnología utilizada es tan avanzada, aún no es infalible. Leer la pantalla LCD puede resultar muy difícil

Llenando el vacío

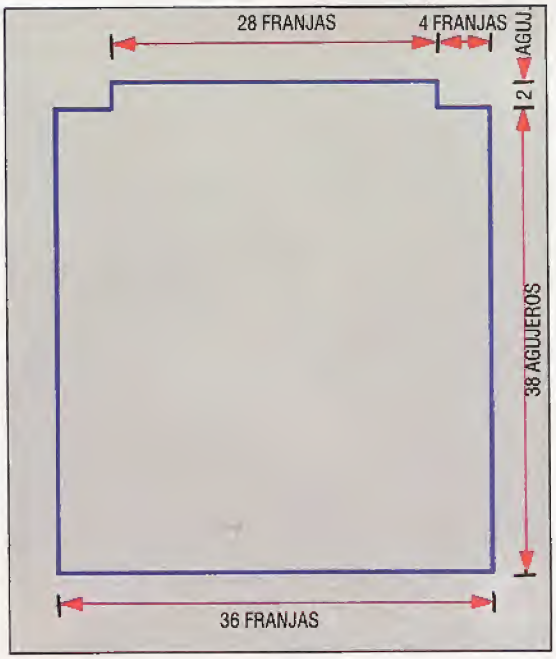
Ahora construiremos la interface diseñada en el capítulo anterior, que le permitirá al usuario del Spectrum controlar el robot

El conector marginal diseñado para el Spectrum habrá de tener anulada la quinta posición desde la izquierda (mirando al conector). Por consiguiente, nuestra primera tarea consiste en insertar un enchufe anulador en el conector marginal; debe enchufarse en la puerta para ampliación. Este enchufe asegura que la interface terminada sólo se pueda insertar en el conector si está orientada correctamente. El enchufe es fácil de construir.

Después de cortar el veroboard de la forma indicada, se puede usar como enchufe anulador una de las esquinas. Primero, pele la pista de cobre y fije el cuadrado cortado de la placa por el borde en la quinta posición del conector marginal (utilizando un poco de supercola), luego corte ambas conexiones de la pantalla. Después efectúe en el veroboard profundos cortes de pistas, usando ya sea una cuchilla o bien una broca espiral de 5 mm. Asegúrese de que cada incisión corte toda la anchura de la pista, ya que, de lo contrario, además de que su interface no funcionaría, incluso se podría dañar el ordenador.

Lista de componentes

Cantidad	Artículo
1	74LS27
1	74LS75
1	74LS126
4	15 K-ohmios, 0,3W
	Carrete alambre estañado de 20 swg
2	Conector DIL de 14 vías
1	Conector DIL de 16 vías
5 m	Cable plano de 20 vías
1	Conector marginal de 2×28 vías y 0,1 pulgadas
1	Conector D de 15 vías
1	Tapa para conector D de 15 vías
1	Veroboard de 36 franjas × 50 aguj.



Observaciones cortantes
Los componentes de la interface para el Spectrum se montan en un trozo de veroboard cortado en la forma que vemos en la ilustración. Use una de las pequeñas esquinas cortadas como enchufe anulador para el conector de la puerta para ampliación. El borde extra en uno de los extremos de la placa aguantará este conector cuando el mismo se suelde en su sitio

Ensamblaje

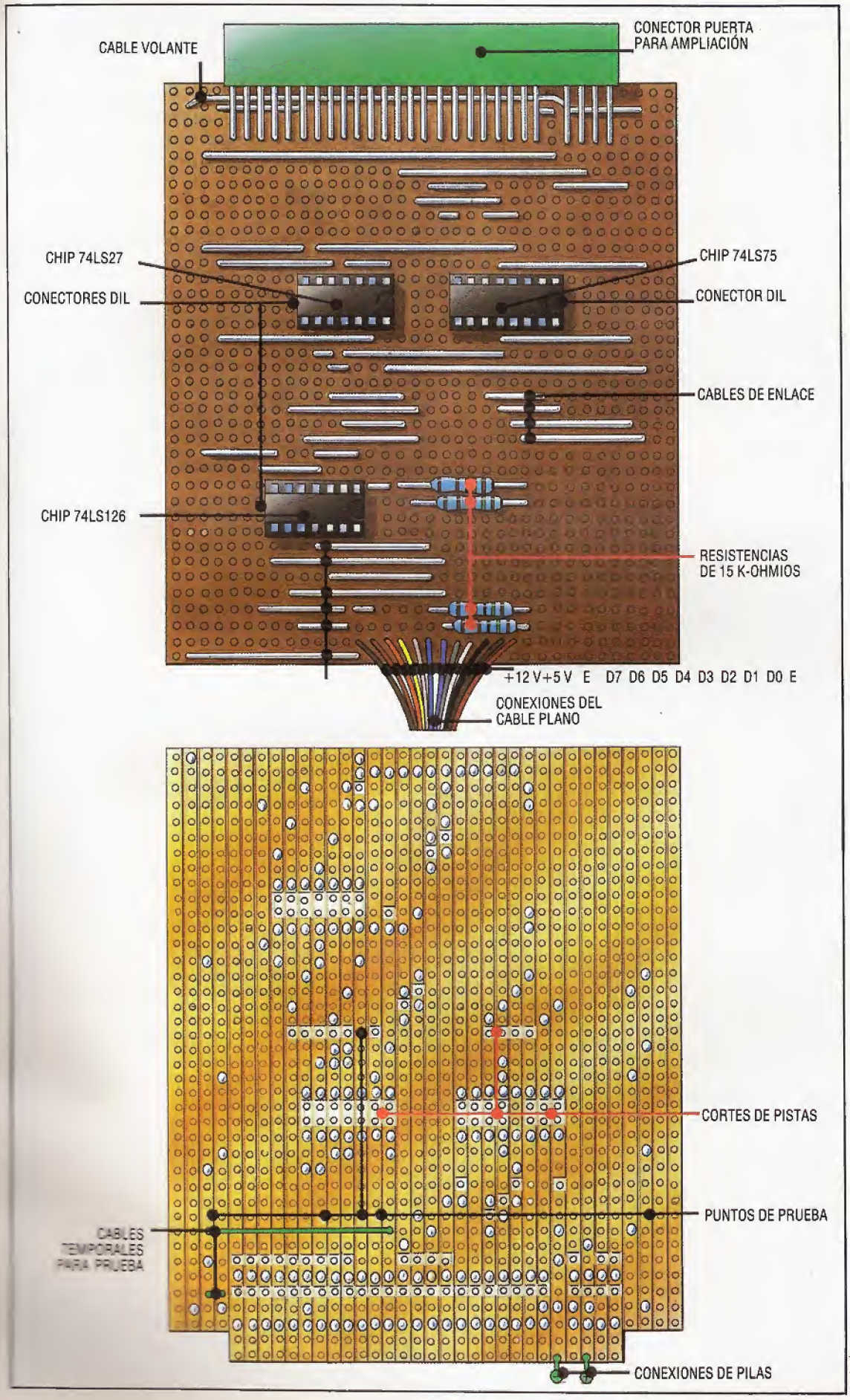
Habiendo efectuado los cortes de pistas, podemos comenzar a ensamblar los componentes en la placa, empezando por los cables de enlace. Éstos se deben cortar holgadamente largos y después insertar con flojedad en la placa. Suelde un extremo y tire del otro hasta tensarlo antes de soldarlo también. Por último, debe cortar bien junto a la placa los trozos de cable sobrantes con un par de cortadores de cable. Cuando todos los cables de enlace se hayan soldado en su sitio, coloque el cable volante aislado, empleando un trocito de uno de los cabos del cable plano.

Después de fijar todos los cables de enlace ya se puede soldar en su sitio el conector marginal. Pero como el sistema de conector marginal en realidad está diseñado para placas de doble cara (como el que utiliza el propio Spectrum) y no para el veroboard, que es de una sola cara, se trata de una operación que tiene su truco. Primero, incline una fila de patillas sobre el conector marginal, como se muestra. Cuando el conector esté posicionado en el tablero, éstas formarán la fila inferior de patillas. Ahora suelde 3 cm de cable estañado en cada una de las de la fila superior. Inserte las patillas inclinadas en la fila de agujeros más próxima al borde de la placa e incline en 90° cada uno de los trozos de cable; insértelos en la otra fila de agujeros (señalados en el diagrama). Luego coloque un poco de supercola en el borde de la placa entre las dos esquinas cortadas y presione el conector marginal en su sitio.

Suelde cuidadosamente cada patilla a las pistas de la placa de franjas. Antes de seguir adelante, compruebe el lado de cobre de la placa y quite, con un cuchillo puntiagudo, cualquier soldadura que pueda estar haciendo puente entre las pistas.

Ahora suelde el resto de los componentes en sus lugares en la placa. Primero se han de soldar los tres conectores DIL y luego las cuatro resistencias de 4 K-ohmios. Por último, inserte los tres chips en su sitio, asegurándose de colocarlos del lado correcto. Las muescas de uno de los extremos de los chips deben apuntar todas en la misma dirección: hacia la derecha cuando se sostiene la placa, con el conector marginal arriba de todo.

Ahora la interface ya está completa. No obstante, antes de enchufarla en el ordenador, es importante comprobarla concienzudamente. Dado que la puerta para ampliación del Spectrum nos otorga ac-



Kevin Jones

Trazado del tablero

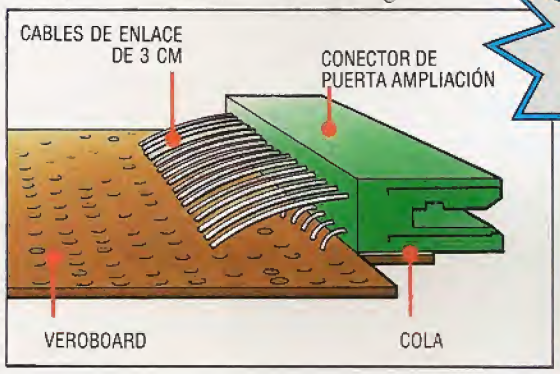
Suelde primero los cables de enlace y resistencias en su sitio en el lado para componentes de la placa, después instale el conector de la puerta para ampliación y los conectores DIL. Asegúrese de que los tres chips estén en la orientación correcta (las muescas deben apuntar hacia el mismo lado). Antes de enchufar a su Spectrum la placa de interface, compare cuidadosamente su trabajo con el diagrama.

En la cara posterior de cobre, asegúrese de que todos los cortes de pistas interrumpan correctamente las pistas, utilizando un medidor para probar la posible continuidad. Los enlaces señalados en verde son conexiones temporales y se deben eliminar una vez realizada con éxito la comprobación

ceso ilimitado a los buses de direcciones y de datos, a las patillas del procesador y a las fuentes de 12 V y 9 V, es importante que, sin habernos percatado de ello, no hayamos conectado estas fuentes de voltaje a la delicada electrónica de los niveles TTL. Primero se debe efectuar una comprobación visual. Repase cuidadosamente toda la placa. Compruebe que no haya puentes de soldadura entre pistas en ningún punto, que todos los componentes y los enlaces de cables estén en el lugar que indican los diagramas y que los chips estén colocados por el lado correcto.

En el próximo capítulo tenderemos el cable plano que conecta el robot a la placa de la interface y veremos versiones para el Spectrum de algunos de los programas que hemos escrito previamente.

Vista lateral
Para poder utilizar un conector marginal estándar para la placa de interface, hemos de efectuar ligeras modificaciones en las patillas. Habiendo eliminado las patillas de la quinta posición, incline el grupo de patillas inferior en ángulo recto y suelde 3 cm de cable estañado al grupo superior. Empuje las patillas a través de los agujeros correspondientes de la placa, comprobando que queden situadas correctamente. La fila inferior de patillas debe situarse en la tercera fila de agujeros y la fila superior, en la sexta. Pegue el conector en su sitio antes de soldar las patillas a la placa



Kevin Jones

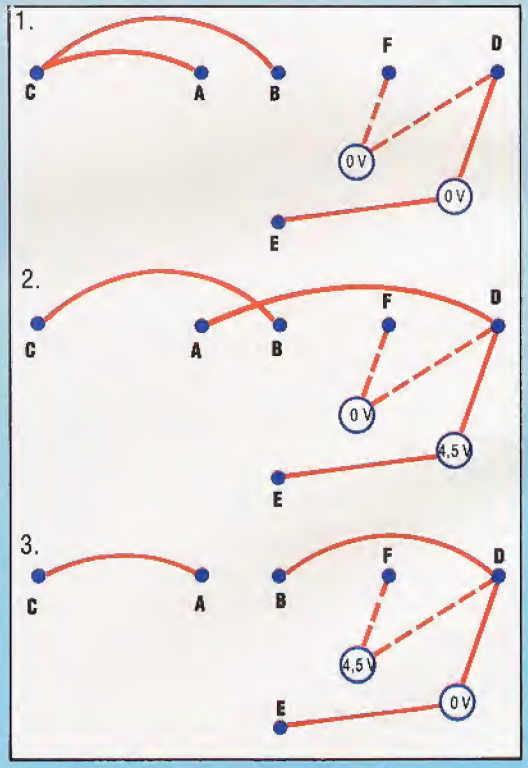
¡ATENCIÓN!

La puerta para ampliación del Spectrum nos permite acceder a la delicada electrónica interna. En la puerta también hay presentes salidas de voltaje que si, inadvertidamente, se dirigieran al bus de datos o al de direcciones, dañarían su ordenador. Compruebe TODAS las conexiones antes de enchufar la interface en la puerta y encenderla

Comprobación de la placa

Antes de enchufar la placa de interface en la puerta para ampliación del Spectrum, podemos realizar una serie de comprobaciones eléctricas para asegurar que la lógica decodificadora esté funcionando correctamente. Para hacerlo necesitamos un tester, una pila de 4,5 V y varios trozos cortos de cable forrado. Primero, realice las dos conexiones que se indican en color verde en el lado de cobre de la placa de circuitos y conecte los terminales de la pila a los puntos adecuados de la placa. Utilice dos trozos de cable forrado para conectar el punto A al punto C y el punto B al punto C. Verifique su trabajo y elimine el exceso de soldadura que pudiera haber entre pistas. Ahora ya podemos comprobar los niveles de voltaje en los puntos E y F. Con la sonda negativa del medidor colocada en el punto D, toque con la sonda positiva primero el punto E y luego el punto F. Estas dos lecturas deben ser de entre 1 V y 0 V. Desuelde el cable entre A y C en C y reconecte el extremo suelto al punto D. Con la sonda negativa de su medidor aún en el punto D, tome nuevas lecturas de E y F. Ahora E debe dar una lectura de 4,5 V y F debe seguir siendo de 0 V. Desconecte el cable de A a D en D, y el cable de B a C en C. Intercambie los extremos sueltos y vuelva a efectuar las conexiones de modo que B se conecte con D y A se conecte con C. Vuelva a tomar lecturas de E y F. Ahora F debe dar 4,5 V y E debe dar 0 V. Si estas pruebas producen las lecturas correctas en los puntos E y F, quite de la placa las conexiones de la pila, los dos cables de enlace verdes y los cables volantes entre A y C, y B y D. Sin embargo, si alguna de estas lecturas no fuera la que indicamos, vuelva a revisar la construcción de toda la placa. La placa de interface no se debe enchufar en el Spectrum bajo ninguna circunstancia hasta que las lecturas del medidor no sean las correctas. En la tabla ofrecemos las diversas conexiones de prueba y las lecturas correctas.

Conexiones	Lecturas
A-C, B-C	E=0 V F=0 V
A-D, B-C	E=4,5 V F=0 V
A-C, B-D	E=0 V F=4,5 V



Este diagrama refleja las conexiones de comprobación a efectuar entre los puntos A, B, C y D, y las lecturas correctas que se deben obtener en los puntos E y F en cada una de las tres etapas de prueba

Fichas archivadas

Consideramos ahora la llamada OSFIND y algunas otras rutinas encargadas de la escritura y recuperación de los datos en su almacenamiento

La OSFIND es llamada en la dirección &FFCE y sirve para abrir un fichero para lo que se llama *acceso al byte*. Es sencillamente la misma tarea de apertura de un fichero en BASIC por medio de OPENOUT, OPENIN o bien OPENUP. También sirve para cerrarlo, a semejanza de CLOSE# en BASIC. El registro A especifica la operación a realizar y los registros X e Y apuntan al nombre del fichero almacenado en memoria.

- *A=0:* Significa que se ha de cerrar el fichero. El registro Y retiene aquí el "número" atribuido al fichero cuando éste se abrió. Si se pone el número del fichero a 0 se cerrarán todos los ficheros actualmente abiertos.
- *A=&40:* Equivale a la orden OPENIN en BASIC. Se abre un fichero sólo para la grabación de datos (bytes).
- *A=&80:* Lo mismo que OPENOUT del BASIC. Se abre un fichero sólo para lectura de datos (bytes).
- *A=&C0:* Equivale al BASIC OPENUP. Esta vez se abre un fichero tanto para escribir como para leer en él. Se puede aplicar a cualquier sistema de ficheros que admita ficheros de acceso al azar.

Al usar OSFIND para abrir un fichero puede que no siempre tengamos éxito. Es posible que haya, por ejemplo, una etiqueta de protección contra escritura en el disco. Si el fichero no puede abrirse, el registro A se pone a cero al volver de OSFIND. Si tenemos éxito y el fichero es abierto, el "número" del fichero queda en A. Este valor se guardará para futuros usos.

Las siguientes llamadas a la ROM de un sistema de ficheros que examinaremos tienen algo en común: todas emplean el número del fichero generado por OSFIND.

La llamada OSARGS

Esta rutina es llamada en la dirección &FFDA y realiza numerosas tareas. Presenta la particularidad de que su bloque de parámetros, tan sólo de cuatro bytes de longitud, se almacena en la página cero. El registro X retiene la dirección del primer byte del bloque de parámetros, el Y retiene el número del fichero y el A especifica la operación que ha de ejecutarse. Existe un caso especial de OSARGS, que es cuando Y está a cero. En este caso la rutina da información sobre el sistema de ficheros empleado más que sobre el fichero en sí, o bien se encarga de que todos los ficheros abiertos en el sistema estén actualizados en relación a los datos que puedan quedar todavía en los buffers destinados a dichos ficheros. Veamos estas dos rutinas primero.

- *Y=0:A=0:* Una llamada OSARGS con estos parámetros y el registro X apuntando al área libre de la página cero produce una información del sistema de ficheros. Los datos se darán en el registro A. Esta llamada OSARGS es la única que funciona en el sistema para cassette. El dato que se da en el registro A se interpreta según el cuadro que ofrecemos al margen.
- *Y=0:A=255:* La otra llamada con Y a cero se produce cuando A retiene el valor 255. Una llamada a OSARGS en estas circunstancias hará que cada archivo quede actualizado con los buffers de la máquina. Si no hay ningún archivo abierto, no tendrá lugar actualización alguna.
- *Y=Número archivo:* Si el registro Y contiene el número del archivo, las operaciones realizadas dependen del contenido de A (como resumimos en el cuadro inferior). El "puntero secuencial" de un fichero abierto indica al sistema operativo dónde se ha de leer o escribir el siguiente byte. Este puntero resulta modificado por la función en BASIC PTR# y por estas rutinas. Como era de esperar, el dato se almacena en el bloque de parámetros de la página cero con su BmnS en la dirección (X+0). Cuando el dato es leído desde el sistema de ficheros por OSARGS, se almacena en el bloque de parámetros para su acceso posterior.

Contenido del reg. A	Descripción
0	Ningún sist. fich.
1	Cinta 1200 baud.
2	Cinta 300 baud.
3	ROM
4	DISCO
5	Red
6	Telesoft

Contenido del registro A	Descripción
0	Lee la posición actual del puntero secuencial en el fichero indicado por el registro Y
1	Escribe el valor del bloque de parámetros en el puntero secuencial. Se parece a la instr. PTR≠n=valor
2	Pone la longitud del archivo en el bloque de parámetros
255	Actualiza arch. en el soporte (el núm. de aquél está contenido en el reg. Y)

De esta exposición de la llàmada resultará claro por qué el bloque de parámetros de OSARGS debe estar en la página cero: no queda más que un registro, el X, para guardar la dirección.

OSBGET y OSBPUT

Estas rutinas se usan, respectivamente, para leer y escribir bytes individuales en un fichero abierto. OSBGET se llama en la dirección &FFD7 y su empleo es muy sencillo. El registro Y se carga con el número de fichero, proporcionado por OSFIND, y llamando a &FFD7 se carga un byte. Este es tomado (leído) de la posición actual del puntero secuencial y devuelto en A.

OSBPUT es la operación inversa; escribe un byte en el fichero en la posición indicada por el puntero secuencial. Para usar OSBPUT, el byte que se ha de escribir en el fichero se sitúa en A, mientras que Y retiene el número del fichero. Seguidamente se hace una llamada a &FFD4 para que ejecute la llamada. La breve rutina siguiente muestra el empleo de OSBPUT para grabar en un fichero un byte individual.

```
LDA   #&80     / preparación para abrir fichero
LDX   # name   MOD 256
LDY   # name   DIV 256
JSR   OSFIND   / abre fichero
STA   &71      / almacena número fichero
TAY            / lo pone además en el reg Y
LDA   #65      / byte a escribir
JSR   OSBPUT   / lo hace
LDA   #0
LDY   &71      / preparación para cerrar fichero
JSR   OSFIND   / cierra fichero
RTS            / fin
```

La llamada OSGBPB

Se llama en la dirección &FFD1. Transfiere grupos de bytes entre la memoria y los ficheros abiertos. Nos puede proporcionar también información sobre el sistema de ficheros en disco, si se está empleando. Aquí sólo nos ceñiremos, sin embargo, a su utilidad en la transferencia de datos. Y a este propósito diremos que la llamada no funciona en un sistema de ficheros en cinta; lo que no debe sorprender demasiado, dado que su utilidad es relevante sólo cuando empleamos ficheros de acceso directo desde lenguaje máquina.

El bloque de los parámetros para la llamada es el que se muestra en el siguiente cuadro, y se apunta a él por medio de los contenidos de X e Y, como es habitual. El registro A especifica la operación a realizar.

Bloque de parámetros para OSGBPB	
Dir.	**Descripción**
0	Núm. fichero, proporcionado por OSFIND
1	BmnS de la dir. del dato por leer o escribir
2	Sin especificar
3	Sin especificar
4	BmsS de la dirección del dato
5	BmnS del número de bytes por trasladar
6	Sin especificar
7	Sin especificar
8	BmsS del número de bytes por trasladar
9	BmnS de la posición del puntero secuencial que hay que usar (permite la selección del lugar donde escribiremos o leeremos dentro del fichero)
10	Sin especificar
11	Sin especificar
12	BmsS de la posición del puntero secuencial

Una vez más advertimos que si usted emplea un BBC Micro estándar, la dirección se especifica por medio del contenido de los bytes 1 y 4 del bloque de parámetros.

Existen cuatro tipos de operaciones de transferencia de datos: dos de lectura y dos de escritura. Una operación de escritura y otra de lectura dan acceso a la información en el fichero en la posición especificada por la entrada del puntero secuencial en el bloque de parámetros mencionado anteriormente, mientras que las otras operaciones de lectura y escritura ignoran esta información por completo y utilizan el puntero secuencial en cualquier posición que se halle dentro del fichero. Así A=1 escribe bytes en el puntero especificado en el bloque; A=2 escribe bytes en el fichero ignorando la entrada del bloque de parámetros; A=3 lee datos del puntero secuencial especificado en el bloque; y A=4 lee bytes pero ignora el bloque de parámetros. El siguiente programa emplea las llamadas OSGBPB para escribir y leer un simple fichero de datos:

```
10 REM solo para sistema en disco
20 DIM block 20
30 DIM data 100
40 $data="Este es un programa de prueba"
50 block!1=data : REM direccion del dato a escribir
60 block!5=100:REM cantidad de datos a escribir
70 block!9=0:REM puntero secuencial=0
80 PRINT "Apertura fichero para lectura"
90 Y%=OPENOUT("FICHERO") : REM abre el fichero
100 block?0=Y%                : REM numero de fichero
110 X%=block MOD 256
120 Y%=block DIV 256
130 A%=1: REM preparacion para escribir datos
140 CALL &FFD1 : REM lo hace
150 CLOSE£(block?0)
160 PRINT "Cierre del fichero"
170 TIME=0 : REPEAT:UNTIL TIME=400
180 PRINT "Apertura fichero para escritura"
190 PRINT "Ahora se leen los datos!"
200 $data=STRING$(100," "):REM borra area datos
210 Y%=OPENIN("FICHERO"): REM abre el fichero
220 block?0=Y%                : REM numero de fichero
230 block!1=data : REM direccion de datos para la operacion de lectura
240 block!5=100 : REM numero de bytes
250 block?9=0 : REM puntero secuencial a cero
260 X%=block MOD 256
270 Y%=block DIV 256
280 A%=3 : REM preparacion para lectura
290 CALL &FFD1 : REM lo hace
300 CLOSE£(block?0)
310 PRINT "Cierra de nuevo el fichero"
320 PRINT $data
330 END
```

El estado del flag de arrastre después de la llamada a &FFD1 indica si se ha logrado la transferencia o no. Si C está a cero entonces la transferencia se realizó con toda normalidad. Si C está a uno, es que ésta falló.

Otra llamada OS que, por último, tiene que ver con operaciones de ficheros es la OSFSC, pero su utilización es mínima a la hora de programar. La llamada es diferente de las otras ya mencionadas aquí dado que no existe una dirección de llamada directa para ella. Hay que llamarla en la dirección contenida en su vector. En los programas en lenguaje máquina, esta llamada puede hacerse con la instrucción

```
JMP    (&21E)
```

Hay otras rutinas OS que cumplen funciones específicas para ciertos sistemas de ficheros. Por ejemplo, las llamadas OSWORD se usan para acceder al chip de control de discos flexibles en un sistema de ficheros en disco. Igualmente, las tareas sencillas del sistema son confiadas a un pareja de llamadas OSBYTE.

Digitaya

En este capítulo de nuestra serie dedicada a programar juegos de aventuras damos el listado completo de "Digitaya"

La estructura de *Digitaya* guarda muchas semejanzas con la de *El bosque encantado*, pero el juego está hecho a una escala mucho mayor. El trazado original para *Digitaya* implicaba un mapa con 100 escenarios que conformaban el trazado interno de un ordenador, junto con memoria, bus de datos, procesador y gran parte del resto del hardware que hay en el interior de un ordenador personal medio. La tarea del jugador consiste en seguir el rastro del misterioso Digitaya, que está cautivo en algún lugar dentro de la máquina. Debe sortear muchos peligros, valiéndose de sus conocimientos sobre el trazado interno de un ordenador para rescatar a Digitaya.

Digitaya utiliza rutinas similares a las de *El bosque encantado* para llevar a cabo las funciones normales del juego, como trasladarse de un escenario a otro, recoger y dejar objetos y examinar los alrededores. No obstante, al esqueleto de esta estructura se le han añadido rutinas especiales para dar cabida a los peligros y trampas especiales del juego.

Cuando se participa en un juego de aventuras, suele ser aconsejable ir trazando un mapa de la ruta con lápiz y papel a medida que uno va atravesando el mundo de la aventura. La tarea primordial de todo jugador de aventuras que se precie consiste en ponerse en el lugar del creador del juego. Recrear el mapa de escenarios a menudo es un buen paso para alcanzar este objetivo. *Digitaya* está diseñado en una cuadrícula plana, pero no espere que todos los mapas de aventuras existan solamente en dos dimensiones. Con la clase de estructura de programa que hemos esbozado en el proyecto, son bastante factibles los mapas tridimensionales. También se podría concebir un juego en cuatro dimensiones, siendo el tiempo la cuarta dimensión. Tal juego, construido en base a un mapa en continuo cambio, probablemente derrotaría a todos los participantes excepto a los más experimentados.

El listado que ofrecemos es para el Commodore 64, aunque el programa se podrá ejecutar en la mayoría de las máquinas con BASIC tipo Microsoft. Se ofrecen complementos para el BBC Micro.

Se pueden suprimir del listado algunas sentencias REM para reducir el esfuerzo de introducción, pero para asegurar que el programa funcione correctamente suprima sólo las REM del final de las líneas de programa que incluyan otro código. Las líneas que sólo contienen sentencias REM por lo general son títulos de subrutinas y sus números de línea se utilizan para las llamadas GOSUB. Es probable que la eliminación de tales líneas provoque un error UNDEFINED STATEMENT (sentencia no definida).

Complementos al BASIC

Spectrum:
Debido a la forma inusual en que el Spectrum manipula las matrices y variables en serie, los complementos para *Digitaya* son tan extensos que sería imposible listarlos aquí. Usted habrá de remitirse a los capítulos anteriores. No obstante, para ayudarlo a recodificar el listado, detallamos los puntos principales a los que atender. Primero, el Spectrum sólo permite variables en serie de un único carácter. Todos los nombres de variables que requieran conversión se ofrecieron ya en una tabla. Asimismo, el Spectrum sólo admite matrices en serie de longitud fija, estableciéndose la longitud de cada elemento de la matriz mediante la sentencia DIM. Ello puede plantear problemas si, pongamos por caso, se dimensionara una matriz en serie de modo que cada elemento tuviera 20 caracteres de largo. Si a un elemento de la matriz se le asignara entonces una serie de 15 caracteres, los cinco caracteres restantes del elemento se rellenarían con espacios en blanco. Hemos de recortar los espacios del elemento de la matriz, por ejemplo, antes de añadírsela a cualquier frase. La variable A$ se emplea para pasar el elemento a una subrutina y debe ser asignada antes de efectuar la llamada a la subrutina. He aquí un ejemplo:

Versión Microsoft:

```
3650 SN$="TU  "+IV$(F,1)+"NO SIRVE DE
     NADA, LA FUERZA SE INTENSIFICA"
```

Versión para el Spectrum:

```
3650 LET S$="TU  ":A$=IV$(F,1):GOSUB
     8500:LETS$=S$+"  NO SIRVE DE
     NADA, LA FUERZA SE INTENSIFICA"
```

El otro problema es el borrado de la pantalla. En la versión Commodore se consigue mediante PRINT CHR$(147). Simplemente reemplace estas sentencias por CLS.

BBC Micro:
En el listado de *Digitaya* reemplace LNS() por LS() y LN por L. Sustituya, además, las siguientes líneas:

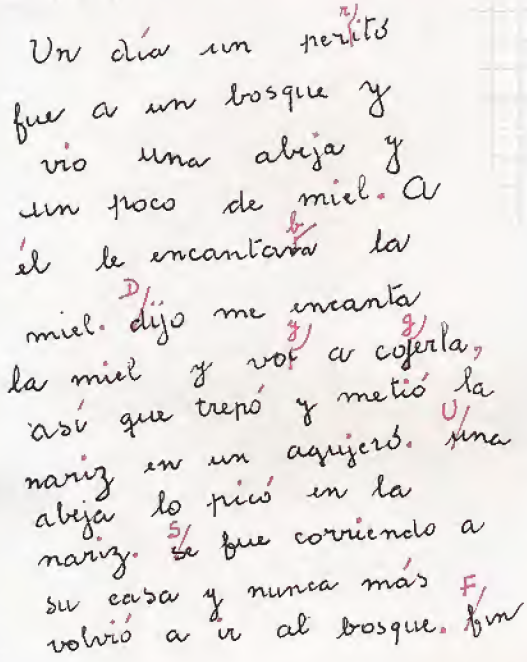
```
1400 A$=GET$
1410 CLS
2630 REPEAT:A$=GET$:UNTIL A$="S" OR
     A$="N"
2750 RA=RND(1)
2820 P=RND(40)+7
3890 RD=RND(1):IF RD>.65 THEN 4110:REM
     ACERTADO
4090 P=RND(40)+7
4520 IV$(4,2)=STR$(RND(40)+7):REM
     REASIGNAR POSICION BILLETE
4570 RN=RND(3)+1
5560 RA=RND(1)
```

La aventura de Digitaya

```
1030 REM **          "DIGITAYA"          **
1040 REM ** UN JUEGO DE AVENTURAS **
1050 REM **      PARA ORDENADOR      **
1090 :
1100 GOSUB6090:REM LEER DATOS MATRIZ
1110 GOSUB1290:REM HISTORIA HASTA AHORA
1120 P=47:REM PUNTO DE PARTIDA
1130 :
1140 REM REM **** AQUI COMIENZA EL BUCLE PRINCIPAL ****
1150 :
1160 MF=0:PRINT
1170 GOSUB1440:REM DESCRIBIR POSICION
1180 GOSUB1560:REM LISTAR SALIDAS
1190 GOSUB 2670:REM ES ESPECIAL P
1200 IF SF=1 THEN 1250:REM SIGUIENTE BUCLE
1210 PRINT:INPUT"INSTRUCCIONES";IS$
1220 GOSUB1700:REM ANALIZAR INSTRUCCIONES
1225 IF F=0 THEN 1210:REM INSTRUCCION NO VALIDA
1230 GOSUB 1900:REM INSTRUCCIONES NORMALES
1240 IF VF=0 THENPRINT"NO COMPRENDO"
1250 IF MF=1 THEN 1160:REM NUEVA POSICION
1260 IF MF=0 THEN 1210:REM NUEVA INSTRUCCION
1270 END
1280 :
1290 REM **** HISTORIA HASTA AHORA ****
1300 SN$="BIENVENIDO A 'DIGITAYA'"
1310 GOSUB5880:REM FORMATEAR
1320 PRINT
1330 SN$="MIENTRAS LA MAQUINA ZUMBA QUEDAMENTE, MIRAS A TU ALREDEDOR."
1340 SN$=SN$+" HACIA EL NORTE Y HACIA EL SUR SE EXTIENDE UN ANCHO BUS."
1350 SN$=SN$+" TU MISION CONSISTE EN ENCONTRAR AL MISTERIOSO DIGITAYA"
1360 SN$=SN$+" Y PONERLO A SALVO SACANDOLO A TRAVES DE UNA DE LAS PUERTAS DE
     SALIDA."
1370 SN$=SN$+" ... PERO, DE CUAL?"
1380 GOSUB 5880
1390 PRINT:PRINT"PULSA UNA TECLA PARA EMPEZAR"
1400 GETA$:IFA$=""THEN1400
1410 PRINTCHR$(147):REM LIMPIAR PANTALLA
1420 RETURN
1430 :
1440 REM **** S/R DESCRIBIR POSICION ****
1450 SN$="TE HALLAS EN "+LN$(P):GOSUB5880
1460 SN$="VES "
1470 REM ** BUSCAR OBJETO **
1480 F=0:SP$=""
1490 FOR I=1 TO 8
1500 IF VAL(IV$(I,2))=P THEN SN$=SN$+SP$+"UN "+IV$(I,1):F=1:SP$=" "
1510 NEXT I
1520 IF F=0 THEN SN$=SN$+"NINGUN OBJETO"
1530 GOSUB5880:REM FORMATEAR
1540 RETURN
1550 :
1560 REM **** S/R LISTAR SALIDAS ****
1570 EX$=EX$(P)
1580 NR=VAL(LEFT$(EX$,2))
1590 EA=VAL(MID$(EX$,3,2))
1600 SO=VAL(MID$(EX$,5,2))
1610 WE=VAL(RIGHT$(EX$,2))
1620 IF(NR OR EA OR SO OR WE)=0THEN RETURN
1630 PRINT:SN$="HAY SALIDAS POR EL "
1640 IF NR<>0 THEN SN$=SN$+"NORTE "
1650 IF EA<>0 THEN SN$=SN$+ "ESTE"
1660 IF SO<>0 THEN SN$=SN$+ "SUR"
1670 IF WE<>0 THEN SN$=SN$+ "OESTE"
1675 GOSUB5880:REM FORMATEAR
1680 PRINT:RETURN
1690 :
1700 REM **** S/R ANALIZAR INSTRUCCION ****
1705 F=0:REM PONER BANDERA A CERO
1710 IFIS$="FIN" OR IS$="LISTAR" THEN VB$=IS$:F=1:RETURN
1720 IF IS$="MIRAR"  THEN VB$=IS$:F=1:RETURN
1730 :
1740 REM ** DESCOMPONER INSTRUCCION **
1750 VB$="":NN$="":REM PONER A CERO VERBO Y SUSTANTIVO
1770 LS=LEN(IS$)
1780 FOR  C=1 TO LS
1790 A$=MID$(IS$,C,1)
1800 IF A$=" " THEN VB$=LEFT$(IS$,C-1):NN$=RIGHT$(IS$,LS-C):F=
     1:C=LS
1810 NEXT
1830 IF F=0 THEN PRINT:PRINT"NECESITO AL MENOS DOS PALABRAS"
1840 RETURN
1850 :
1900 REM **** S/R ACCIONES NORMALES ****
1910 VF=0
1920 PRINT
1930 IF VB$="IR" OR VB$="AVANZAR"THEN VF=1:GOSUB2000
1940 IF VB$="RECOGER" ORVB$="COGER" THEN VF=1:GOSUB2140
1950 IF VB$="DEJAR" ORVB$="PONER" THENVF=1:GOSUB2360
1960 IF VB$="LISTAR"ORVB$="INVENTARIO"THEN VF=1:GOSUB2540
1965 IF VB$="MIRAR" THEN VF=1:MF=1:RETURN
1970 IF VB$="FIN"ORVB$="TERMINAR" THEN VF=1:GOSUB2610
1980 RETURN
1990 :
2000 REM **** S/R DE MOVIMIENTO ****
2010 MF=1:REM ESTABLECER BANDERA DE MOVIMIENTO
2015 GOSUB8600:REM BUSCAR DIRECCION
2020 DR$=LEFT$(NN$,1)
2030 IFDR$<>"N"ANDDR$=[illegible]"S"ANDDR$<>"0"THEN2100
2040 IF DR$="N" AND NR<>0 THEN P=NR:RETURN
2050 IF DR$="S" AND SO<>0 THEN P=SO:RETURN
2060 IF DR$="E" AND EA<>0 THEN P=EA:RETURN
2070 IF DR$="0" AND WE<>0 THEN P=WE:RETURN
2080 PRINT"NO PUEDES ";IS$
2090 MF=0:RETURN
2100 REM EL NOMBRE NO ES OK
2110 PRINT"QUE ES ";NN$;" ?"
2120 MF=0:RETURN
2130 :
2140 REM **** S/R RECOGER ****
2145 IV$(4,1)="BILLETE AL TRIESTADO"
2150 GOSUB5730:REM ES VALIDO EL OBJETO
2160 IF F=0 THEN PRINT"NO HAY NINGUN ";W$:RETURN
2170 REM ** YA HA RECOGIDO EL OBJETO ? ****
2180 OV=F:GOSUB5830
2190 IFHF=1 THEN SN$="YA TIENES EL "+IV$(F,1):GOSUB5880:RETURN
2200 :
2210 REM ** ESTA AQUI EL OBJETO **
2220 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+"NO ESTA AQUI":GOSUB5880: RETURN
2230 :
2240 REM ** AÑADIR OBJETO A LA LISTA **
2250 AF=0:FOR J=1 TO 4
2260 IFIC$(J)=""THENIC$(J)=IV$(F,1):AF=1:J=4
2270 NEXTJ
2280 :
2290 REM ** COMPROBAR SI CUOTA CUBIERTA **
2300 IF AF=0THENPRINT"YA TIENES 4 OBJETOS":RETURN
2310 :
2320 SN$="COGES EL "+IV$(F,1):GOSUB5880
2330 IV$(F,2)="-":REM SUPRIMIR ENTRADA POSICION
2340 RETURN
2350 :
2360 REM ** S/R DEJAR **
2370 GOSUB5730:REM ES VALIDO EL OBJETO
2380 IF F=0 THEN PRINT"NO HAY NINGUN ";NN$:RETURN
2390 :
2400 REM ** LLEVA CONSIGO EL OBJETO ? **
2410 OV=F:GOSUB5830
2420 IFHF=0THENPRINT"TU NO TIENES EL ";IV$(F,1):RETURN
2430 :
2440 REM ** DEJAR OBJETO **
2450 SN$="DEJAS EL "+IV$(F,1):GOSUB5880
2460 IV$(F,2)=STR$(P):REM ACTUALIZAR POSICION OBJETO
2470 :
2480 REM ** SUPRIMIRLO DE LISTA DE OBJ TRANSPORTADOS **
2490 FOR J=1TO4
2500 IF IC$(J)=IV$(F,1)THEN IC$(J)="":J=4
2510 NEXTJ
2520 RETURN
2530 :
2540 REM **** S/R LISTAR INVENTARIO ****
2550 PRINT"OBJETOS QUE LLEVAS CONTIGO:"
2560 FORI=1TO4
2570 PRINT"  ";IC$(I)
2580 NEXTI
2590 RETURN
2600 :
2610 REM **** S/R FIN DEL JUEGO ****
2620 PRINT:PRINT"ESTAS SEGURO (S/N) ?"
2630 GETA$:IFA$<>"S" AND A$<>"N"THEN2630
2640 IFA$="N"THEN RETURN
2650 END
2660 :
2670 REM **** S/R ES ESPECIAL P ****
2680 SF=0:REM QUITAR BANDERA DE ESPECIAL
2690 IF P=37 THEN2780:REM TABLA DE VECTORES
2700 IF P>7 THEN 2750:REM BICHO AL AZAR
2710 ON P GOSUB 2850,2960,3450,3830,4180,4550,5150
2720 RETURN
2730 :
2740 REM ** BICHO AL AZAR **
2750 RA=RND(TI)
2760 IF RA<0.05 THEN GOSUB 5420:REM BICHO
2770 RETURN
2780 REM ** TABLA DE VECTORES **
2790 SF=1
2800 SN$="AVANZAS A GRAN VELOCIDAD HASTA UN NUEVO ESCENARIO":GOSUB5880
2810 FORJ=1TO1000:NEXT:REM PAUSA
2820 P=INT(RND(TI)*40+7)
2830 MF=1:RETURN
2840 :
2850 REM **** S/R TOMA DEL TELEVISOR ****
2860 SF=1
2870 SN$="HAS ENTRADO EN LA TOMA DEL TELEVISOR Y NO TIENES SALIDA."
2880 SN$=SN$+"ESTAS CONDENADO PARA SIEMPRE A SER EL INVITADO DE UN PROGRAMA DE
     TERTULIA DE TV"
2890 GOSUB5880:REM FORMATEAR SALIDA
2900 PRINT
2910 PRINT"BIENVENIDOS AL SHOW....."
2920 FORJ=1TO500:NEXTJ
2930 GOTO 2910
2940 END
2950 :
2960 REM **** S/S PUERTA PARA EL USUARIO ****
2970 SF=1
2980 SN$="PUEDES ESCAPAR PERO EL VENDEDOR DE BILLETES DEL RDD"
2990 SN$=SN$+" TE IMPIDE EL PASO. TE DICE QUE TIENE INSTRUCCIONES"
3000 SN$=SN$+" DE ACEPTAR SOLO ENTRADAS. SIN EMBARGO ACEPTA TODAS LAS
     PRINCIPALES"
```

```
3010 SN$=SN$+ "TARJETAS DE CREDITO."
3020 GOSUB 5880:REM FORMATEAR IMPRESION
3030 :
3040 PRINT:INPUT"INSTRUCCIONES";IS$
3050 GOSUB1700:REM ANALIZAR INSTRUCCIONES
3060 GOSUB1900: REM ACCIONES NORMALES
3070 IF MF=1 THEN RETURN:REM IRSE
3080 IF VF=1 THEN3040:REM SIGUIENTE INSTRUCCION
3090 IF VB$<>"DAR" THENPRINT"NO COMPRENDO":GOTO3040
3100 :
3110 REM ** LA INSTRUCCION ES DAR **
3120 GOSUB5730:REM ES VALIDO EL OBJETO
3130 IFF=0THENPRINT"NO HAY NINGUN ";NN$:GOTO3040:REM SIGUIENTE INSTRUCCION
3140 :
3150 REM ** ES EL OBJ TARJETA DE CREDITO **
3160 IF F<>5 THEN PRINT"EL SOLO ACEPTA TARJETAS DE CREDITO":GOTO3040
3170 :
3180 REM ** LLEVA LA TARJETA **
3190 OV=5:GOSUB5830
3200 IFHF=0THENPRINT"TU NO TIENES LA ";IV$(5,1):GOTO 3040
3210 :
3220 SN$="EL VENDEDOR COGE LA TARJETA Y DICE 'ESTA SERVIRO, SEÑOR'"
3230 GOSUB5880:REM FORMATEAR SALIDA
3240 SN$="SE TE PERMITE ATRAVESAR LA BARRERA Y ENTRAR EN LA PUERTA PARA EL
     USUARIO"
3250 GOSUB5880:REM FORMATEAR SALIDA
3260 :
3270 REM ** LLEVA A DIGITAYA **
3280 OV=6:GOSUB5830
3290 IF HF=1 THEN 3380:REM EXITO
3300 :
3310 REM ** FRACASO **
3320 SN$="BIEN HECHO.HAS CONSEGUIDO ESCAPAR DE LAS GARRAS"
3330 SN$=SN$+" DE LA MAQUINA, PERO HAS FRACASADO EN TU MISION"
3340 SN$=SN$+" DE TRAER DE REGRESO AL MISTERIOSO DIGITAYA"
3350 GOSUB5880:REM FORMATEAR SALIDA
3360 END
3370 :
3380 REM ** EXITO **
3390 SN$="FELICITACIONES, HAS TENIDO EXITO EN TU MISION"
3400 SN$=SN$+" CONSISTENTE EN RESCATAR AL MARAVILLOSO DIGITAYA DE LAS"
3410 SN$=SN$+" GARRAS DE LA MAQUINA."
3420 GOSUB5880:REM FORMATEAR SALIDA
3430 END
3440 :
3450 REM **** S/R PUERTA PARA CASSETTE ****
3460 SF=1
3470 SN$="SIENTES QUE UNA FUERZA IRRESISTIBLE TE EMPUJA HACIA"
3480 SN$=SN$+" LA SUSPENSION MAGNETICA PERMANENTE"
3490 GOSUB5880:REM FORMATEAR
3500 NS=0:REM COMENZAR A CONTAR INSTRUCCIONES
3510 REM ** INSTRUCCIONES **
3520 NS=NS+1:IFNS>3THEN3770:REM ABSORBIDO
3530 PRINT:INPUT"INSTRUCCIONES";IS$
3540 GOSUB1700:REM ANALIZAR INSTRUCCIONES
3550 GOSUB1900:REM ACCIONES NORMALES
3560 IF MF=1 THEN MF=0:PRINT"NO PUEDES MOVERTE...TODAVIA":GOTO3510
3570 IFVF=1THEN3510:REM SIGUIENTE INSTRUCCION
3580 IFVB$<>"USAR"THENPRINT"NO COMPRENDO":GOTO3510
3590 REM ** LA INSTRUCCION ES USAR **
3600 GOSUB5730:REM ES VALIDO EL OBJETO
3610 IFF=0THENPRINT"NO HAY NINGUN ";NN$:GOTO3510
3620 :
3630 REM ** ES EL OBJETO EL ACTIVADOR DEL BUFFER **
3640 IF F=8 THEN 3680:REM OK
3650 SN$="TU "+IV$(F,1)+" NO SIRVE DE NADA, LA FUERZA AUMENTA"
3660 GOSUB 5880:GOTO3510:REM SIGUIENTE INSTRUCCION
3670 :
3680 OV=8:GOSUB5830:REM LLEVA EL ACT DEL BUFFER
3690 IF HF=0THENSN$="NO LLEVAS CONTIGO EL "+IV$(8,1):GOSUB5880:GOTO 3510
3700 :
3710 REM ** SALVADO **
3720 SN$="USAS EL ACTIVADOR DEL BUFFER PARA CONTRARRESTAR EL EMPUJE"
3730 SN$=SN$+" HACIA EL OLVIDO MAGNETICO. LA FUERZA DISMINUYE"
3740 GOSUB5880:REM FORMATEAR
3750 RETURN
3760 :
3770 REM ** ABSORBIDO **
3780 SN$="LA FUERZA SE HACE DEMASIADO INTENSA Y ERES ARROJADO"
3790 SN$=SN$+" A TRAVES DE LA PUERTA PARA CASSETTE A LA NADA MAGNETICA."
3800 GOSUB5880:REM FORMATEAR
3810 END
3820 :
3830 REM **** PUERTA PARA PALANCA DE MANDOS ****
3840 SF=1
3850 SN$="UN USUARIO DE OJOS ENROJECIDOS DISPARA REPETIDAMENTE SU LASER CONTRA
     TI."
3860 GOSUB5880:REM FORMATEAR
3870 :
3880 REM ** INSTRUCCIONES **
3890 RD=RND(TI):IF RD>.65 THEN 4110:REM ACERTADO
3900 PRINT:INPUT"INSTRUCCIONES";IS$
3910 GOSUB1700:GOSUB1900:REM ANALIZAR INSTRUCCION
3920 IFMF=1THENMF=0:PRINT"NO PUEDES MOVERTE...TODAVIA":GOTO3880
3930 IFVF=1"THEN3880:REM SIGUIENTE INSTRUCCION
3940 IF VB$<>"USAR" THENPRINT"NO COMPRENDO":GOTO3880
3950 GOSUB5730:REM ES VALIDO EL OBJETO
3960 IFF=0"THENPRINT"NO HAY NINGUN ";NN$:GOTO3880:REM SIGUIENTE INSTRUCCION
3970 :
3980 REM ** ES EL OBJETO EL ESCUDO LASER **
3990 IFF=3 THEN4000:REM OK
4000 SN$="TU "+IV$(F,1)+" NO SIRVE":GOSUB5880:GOTO3880
4010 :
4020 OV=3:GOSUB5830:REM LLEVA CONSIGO EL ESCUDO LASER
4030 IF MF=0 THENSN$="TU NO TIENES EL "+IV$(3,1):GOSUB5880:GOTO3880
4040 :
4050 REM ** SALVADO **
4060 SN$="USAS EL ESCUDO LASER PARA PROTEGERTE, UNA DESCARGA TE ARROJA"
4070 SN$=SN$+" FUERA DE LA PUERTA PARA PALANCA DE MANDOS Y VUELVES A ENTRAR EN
     LA MAQUINA."
4080 GOSUB5880:REM FORMATEAR
4090 P=INT(RND(TI)*40+7):MF=1:RETURN
4100 :
4100 REM ** ACERTADO **
4120 SN$="ERES ALCANZADO POR EL LASER Y APENAS SI ERES CONSCIENTE DE"
4130 SN$=SN$+" QUE TUS ATOMOS SE HAN DISPERSADO POR LAS CUATRO ESQUINAS"
4140 SN$=SN$+" DEL UNIVERSO"
4150 GOSUB5880:REM FORMATEAR
4160 END
4170 :
4180 REM **** S/R DISPOSITIVO TRIESTADO ****
4190 SF=1
4200 SN$="UN GRAN CARTEL REZA 'E/S POR AQUI' PERO CUANDO TE ACERCAS"
4210 SN$=SN$+" UN REVISOR GRITA 'BILLETES POR FAVOR'"
4220 GOSUB5880:REM FORMATEAR
4230 :
4240 REM ** INSTRUCCIONES **
4250 PRINT:INPUT"INSTRUCCIONES";IS$
4260 GOSUB1700:GOSUB1900:REM ANALIZAR
4270 IFMF=1 THEN RETURN
4280 IFVF=1THEN4240:REM SIGUIENTE INSTRUCCION
4290 IFVB$<> "DAR" AND VB$<>"OFRECER"THENPRINT"NO COMPRENDO":GOTO4240
4300 REM ** LA INSTRUCCION ES DAR **
4310 GOSUB5730:REM ES VALIDO EL OBJETO
4320 IFF=0THENPRINT"NO HAY NINGUN ";NN$:GOTO4240:REM SIGUIENTE INSTRUCCION
4330 :
4340 REM ** ES EL OBJETO EL BILLETE **
4350 IF F=4 THEN 4400:REM OK
4360 SN$="EL REVISOR SACUDE LA CABEZA Y DICE"
4370 SN$=SN$+" 'NO PUEDO ACEPTAR ESTO "+IV$(F,1)
4380 GOSUB5880:GOTO4240:REM SIGUIENTE INSTRUCCION
4390 :
4400 OV=4:GOSUB5830:REM LLEVA CONSIGO EL BILLETE
4410 IFHF=0THENPRINT"TU NO TIENES EL BILLETE":GOTO 4240
4420 :
4430 REM ** OK **
4440 SN$="EL REVISOR ACEPTA TU BILLETE Y TE PERMITE"
4450 SN$=SN$+" ATRAVESAR LA BARRERA."
4460 GOSUB5880:REM FORMATEAR
4470 REM ** SUPRIMIR BILLETE DE LA LISTA **
4480 F=0
4490 FORJ=1TO4
4500 IF IC$(J)=IV$(4,1)THENIC$(J)="":J=4
4510 NEXT J
4520 IV$(4,2)=STR$(INT(RND(TI)*40+8)):REM REASIGNAR POSICION BILLETE
4530 P=15:MF=1:RETURN
4540 :
4550 REM **** ALU ****
4560 SF=1
4570 RN=INT(RND(TI)*3+1)
4580 IF RN=1 THEN CD$="AND"
4590 IF RN=2 THEN CD$="OR"
4600 IF RN=3 THEN CD$="NOT"
4610 SN$="MONTADOS EN LA PARED HAY TRES BOTONES MARCADOS"
4620 SN$=SN$+" 'AND', 'OR' Y 'NOT'. PUEDES GANAR ACCESO AL"
4630 SN$=SN$+" ACUMULADOR PULSANDO EL BOTON CORRECTO"
4640 GOSUB5880:REM FORMATEAR
4650 :
4660 REM ** INSTRUCCIONES **
4670 PRINT:INPUT"INSTRUCCIONES";IS$
4680 GOSUB1700:GOSUB1900:REM ANALIZAR
4690 IF MF=1 THEN RETURN:REM IRSE
4700 IF VF=1 THEN 4670:REM SIGUIENTE INSTRUCCION
4710 IF VB$="USAR" OR VB$="PULSAR" THEN4740
4720 PRINT"NO COMPRENDO":GOTO4670
4730 :
4740 REM ** INSTRUCCION VALIDA **
4750 IF VB$="PULSAR"THEN 4930
4760 REM ** LA INSTRUCCION ES USAR **
4770 GOSUB5730:REM ES VALIDO EL OBJETO
4780 IFF=0THENPRINT"NO HAY NINGUN ";NN$:GOTO4670:REM SIGUIENTE INSTRUCCION
4790 :
4800 REM ** ES EL OBJETO EL LIBRO DE CODIGOS **
4810 IF F=7 THEN 4850:REM OK
4820 SN$="TU "+ IV$(F,1)+" NO SIRVE":GOSUB5880
4830 GOTO4670:REM SIGUIENTE INSTRUCCION
4840 :
4850 OV=7:GOSUB5830:REM LLEVA CONSIGO EL LIBRO DE CODIGOS
4860 IFHF=1THEN4900:REM OK LO LLEVA
4870 SN$="TU NO TIENES EL "+IV$(7,1)
4880 GOSUB5880:GOTO4670:REM SIGUIENTE INSTRUCCION
4890 :
4900 SN$="ABRES EL LIBRO DE CODIGOS Y EN SU INTERIOR ENCUENTRAS ESCRITA LA
     PALABRA '"+CD$+"'"
4910 GOSUB5880:GOTO4670:REM SIGUIENTE INSTRUCCION
4920 :
4930 REM ** LA INSTRUCCION ES PULSAR **
4940 IFNN$="AND"OR NN$="OR"OR NN$="NOT" THEN 4970
4950 SN$="NO HAY NINGUN "+NN$:GOSUB5880:GOTO4670:REM SIGUIENTE INSTRUCCION
4960 :
4970 REM ** CORRECTO O INCORRECTO **
4980 IFNN$=CD$ THEN GOSUB5100:RETURN
```

```
4990 GOSUB5010:RETURN
5000 :
5010 REM **** S/R INCORRECTO ****
5020 SN$="INCORRECTO, SE ABRE UNA PUERTA FALSA Y TE ENCUENTRAS OTRA"
5030 SN$=SN$+" VEZ EN LA MEMORIA PRINCIPAL"
5040 GOSUB5880:REM FORMATEAR
5050 IF RN=1 THEN P=39
5060 IF RN=2 THEN P=35
5070 IF RN=3 THEN P=29
5080 MF=1:RETURN
5090 :
5100 REM ** S/R CORRECTO **
5110 SN$="SE ABRE LA PUERTA QUE DA AL ACUMULADOR Y"
5120 SN$=SN$+" LA CRUZAS":GOSUB5880
5130 P=30:MF=1:RETURN
5140 :
5150 REM **** S/R PUERTA A LA MEMORIA ****
5160 SF=1
5170 SN$="TE SALUDA UN CONSERJE PERO TE DICE QUE NO TE PUEDE DEJAR PASAR"
5180 SN$=SN$+" A MENOS QUE LE DES UNA DIRECCION":GOSUB5880
5190 REM ** INSTRUCCIONES **
5200 PRINT:INPUT"INSTRUCCIONES";IS$
5210 GOSUB1700:GOSUB1900:REM ANALIZAR
5220 IF MF=1 THEN RETURN:REM IRSE
5230 IF VF=1 THEN 5200:REM SIGUIENTE INSTRUCCION
5240 IF VB$<>"DAR"THEN PRINT"NO COMPRENDO":GOTO5200
5250 :
5260 GOSUB5730:REM ES VALIDO EL OBJETO
5270 IFF=0THENPRINT"NO HAY NINGUN ":NN$:GOTO5200:REM SIGUIENTE INSTRUCCION
5280 :
5290 REM ** ES EL OBJETO DIRECCION **
5300 IF F=1 THEN5330:REM OK
5310 PRINT"NECESITA TU DIRECCION":GOTO5200
5320 :
5330 OV=1:GOSUB5830:REM LLEVA LA DIRECCION
5340 IF HF=1 THEN 5370
5350 SN$="TU NO TIENES LA "+IV$(1,1):GOSUB5880:GOTO5200
5360 :
5370 REM ** OK PASAR **
5380 SN$="EL CONSERJE EXAMINA TU DIRECCION Y TE PERMITE"
5390 SN$="SN$+" PASAR":GOSUB5880
5400 P=40:MF=1:RETURN
5410 :
5420 REM **** BICHO AL AZAR ****
5430 SF=1
5440 SN$="POR DETRAS DE UN CHIP APARECE UN BICHO ENORME Y ASQUEROSO"
5450 SN$="SN$+" Y ARREMETE CONTRA TI":GOSUB5880
5460 :
5470 REM ** INSTRUCCIONES **
5480 PRINT:INPUT"INSTRUCCIONES";IS$
5490 GOSUB1700:GOSUB1900:REM ANALIZAR
5500 IFMF=1THENMF=0:PRINT"NO PUEDES MOVERTE...TODAVIA":GOTO5480
5510 IFVF=1THEN5480:REM SIGUIENTE INSTRUCCION
5520 IF VB$="MATAR" OR VB$="LUCHAR" THEN5550
5530 PRINT"NO COMPRENDO":GOTO5480
5540 :
5550 REM ** LA INSTRUCCION ES LUCHAR/MATAR **
5560 RA=RND(TI)
5570 IFRA<0.5THENGOSUB5600
5580 GOSUB5670:RETURN
5590 :
5600 REM **** S/R MUERTO ****
5610 SN$="LUCHAS CON EL BICHO. TE ARROJA UNA LLUVIA DE ERRORES"
5620 SN$=SN$+" QUE SE TE METEN EN EL CEREBRO."
5630 SN$=SN$+" FINALMENTE TU CABEZA YA NO RESISTE MAS Y ESTALLA."
5640 GOSUB5880
5650 END
5660 :
5670 REM **** S/R LO MATAS ****
5680 SN$="LUCHAS CON EL BICHO Y AUNQUE LA PELEA ES DURA"
5690 SN$=SN$+" FINALMENTE LOGRAS VENCERLO Y SOBREVIVIR.":GOSUB5880
5700 RETURN
5710 :
5720 :
5730 REM **** S/R OBJETO VALIDO ****
5740 NN$=NN$+" ":LN=LEN(NN$):F=0:C=1
5745 FOR K=1 TO LN
5750 IF MID$(NN$,K,1)<>" " THENNEXT K:RETURN
5755 W$=MID$(NN$,C,K-C):C=K+1:LW=LEN(W$)
5760 FORJ=1 TO 8
5770 LI=LEN(IV$(J,1)):REM LONGITUD OBJETO
5780 FORI=1 TO LI
5790 IFMID$(IV$(J,1),I,LW)=W$THENF=J:I=LI:J=8:K=LN
5800 NEXT I,J,K
5810 RETURN
5820 :
5830 REM **** S/R LLEVA CONSIGO OBJETO ****
5840 MF=0
5850 IFIV$(OV,2)="-1"THEN HF=1
5860 RETURN
5870 :
5880 REM **** S/R FORMATEAR IMPRESION ****
5890 LC=0:      REM CONTADOR CAR/LINEA
5900 OC=1:      REM CONTADOR ANTIGUO
5910 OW$="":    REM PALABRA ANTIGUA
5920 LL=40:     REM LONGITUD LINEA PANTALLA
5930 SN$=SN$+" FICTICIA"
5940 PRINT
5950 FOR C=1 TO LEN(SN$)
5960 LC=LC+1
5970 IF MID$(SN$,C,1)=" " THENGOSUB6020
5980 NEXTC
5990 PRINT
6000 RETURN
6010 :
6020 REM **** S/R COMPROBACION FINAL LINEA ****
6030 NW$=MID$(SN$,OC,C-OC+1)
6040 IF LC<LL THENPRINTOW$;:GOTO6060
6050 PRINTOW$:LC=LEN(NW$)
6060 OC=C+:OW$=NW$
6070 RETURN
6080 :
6090 REM **** S/R LEER DATOS MATRIZ ****
6100 REM ** LEER INVENTARIO **
6110 DIM IV$(8,2),IC$(4)
6120 FOR C=1TO8
6130 READ IV$(C,1),IV$(C,2)
6140 NEXT C
6150 :
6160 REM ** LEER DATOS ESCENARIO Y SALIDAS **
6170 DIM LN$(55),EX$(55)
6180 C1=0:C2=0:REM INICIALIZAR SUMAS DE CONTROL
6190 FOR C=1 TO 54
6200 READ LN$(C),EX$(C)
6210 C1=C1+VAL(LEFT$(EX$(C),4))
6220 C2=C2+VAL(RIGHT$(EX$(C),4))
6230 NEXT C
6240 READ CA:IF CA<>C1 THEN PRINT"ERROR EN SUMA DE CONTROL":STOP
6250 READ CB:IF CB<>C2 THEN PRINT"ERROR EN SUMA DE CONTROL":STOP
6260 RETURN
6270 REM **** DATOS INVENTARIO ****
6280 DATA NUMERO DE DIRECCION,45,LLAVE,34,ESCUDO LASER,25
6290 DATA BILLETE AL TRIESTADO,26,TARJETA DE CREDITO DATOS,28
6300 DATA DIGITAYA,30,LIBRO DE CODIGOS,19,DISPOSITIVO ACTIVADOR DEL BUFFER,13
6310 :
6320 REM **** DATOS ESCENARIOS Y SALIDAS ****
6330 DATA EN LA TOMA DEL TELEVISOR,00000000
6340 DATA EN LA PUERTA PARA EL USUARIO,00090100
6350 DATA EN LA PUERTA PARA CASSETTE,00110000
6360 DATA EN LA PUERTA PARA PALANCA DE MANDOS,00130000
6370 DATA EN UN DISPOSITIVO TRIESTADO,00170000
6380 DATA EN LA UNIDAD ARITMETICO LOGICA,00310016
6390 DATA EN LA PUERTA DE ACCESO A LA MEMORIA,00490000
6400 DATA EN EL BUS DE E/S,09000001
6410 DATA EN EL BUS DE E/S,10000802
6420 DATA EN EL BUS DE E/S,11000900
6430 DATA EN EL BUS DE E/S,12001003
6440 DATA EN EL BUS DE E/S,13531100
6450 DATA EN EL BUS DE E/S,14001204
6460 DATA EN EL BUS DE E/S,15001300
6470 DATA EN EL BUS DE E/S UN CARTEL REZA 'S OUT H',00001400
6480 DATA EN EL REGISTRO DE DATOS,00061700
6490 DATA EN UN BUS DE 8 PISTAS,16001805
6500 DATA EN UN BUS DE 8 PISTAS,17001900
6510 DATA EN UN BUS DE 8 PISTAS,18002000
6520 DATA EN EL BUS DE 8 PISTAS,19292100
6530 DATA EN EL BUS DE 8 PISTAS,20282200
6540 DATA EN EL BUS DE 8 PISTAS,21272300
6550 DATA EN UN BUS DE 8 PISTAS,22262400
6560 DATA EN UN BUS DE 8 PISTAS,23250000
6570 DATA EN LA MATRIZ DE CARACTERES,26360024
6580 DATA EN LA PARTE SUPERIOR DE LA MEMORIA,27352523
6590 DATA EN LA MITAD DE LA MEMORIA,28342622
6600 DATA EN LA MITAD DE LA MEMORIA,29332721
6610 DATA EN LA PARTE INFERIOR DE LA MEMORIA,00542820
6620 DATA EN LA GUARIDA DEL ACUMULADOR, 00000600
6630 DATA EN UN LARGO CORREDOR,00420006
6640 DATA EN UN REGISTRO DE INDICE,31000000
6650 DATA EN LA PARTE INFERIOR DE LA MEMORIA,54403428
6660 DATA EN LA MITAD DE LA MEMORIA,33393527
6670 DATA MUY ARRIBA EN LA MEMORIA,34383626
6680 DATA EN LA MATRIZ DE CARACTERES,35370025
6690 DATA EN UNA TABLA DE VECTORES ALEATORIOS,00000000
6700 DATA EN LA PARTE SUPERIOR DE LA MEMORIA CONTEMPLANDO UN BUS DESDE
     ARRIBA,39003735
6710 DATA EN LA MITAD DE LA MEMORIA,4003834
6720 DATA EN LA MEMORIA -- HACIA EL ESTE HAY UNA PUERTA,41003933
6730 DATA EN LA PARTE INFERIOR DE LA MEMORIA,00004054
6740 DATA EN UN CORREDOR,00430031
6750 DATA EN UN CORREDOR,00440042
6760 DATA EN UN CORREDOR,00004543
6770 DATA EN EL REGISTRO DE DIRECCIONES,00004600
6780 DATA EN UN BUS DE 16 PISTAS,45004700
6790 DATA EN UN BUS DE 16 PISTAS,46004800
6800 DATA EN UN BUS DE 16 PISTAS,47004900
6810 DATA EN UN BUS DE 16 PISTAS SE VISLUMBRA
     HACIA EL OESTE UNA GRAN PUERTA 48005007
6820 DATA EN UN BUS DE 16 PISTAS,49005100
6830 DATA EN UN BUS DE 16 PISTAS,5005200
6840 DATA EN UN BUS DE 16 PISTAS,51000000
6850 DATA EN UN VECTOR A LA MEMORIA,00290012
6860 DATA EN LA PARTE INFERIOR DE LA MEMORIA,00413329
6870 REM ** DATOS SUMA DE CONTROL **
6880 DATA 100169,103973
8599 :
8600 REM **** S/R BUSCAR DIRECCION ****
8610 NN$=NN$+" ":LN=LEN(NN$):C=1
8620 FORI=1 TO LN
8630 IF MID$(NN$,I,1)<>" " THEN NEXT I:RETURN
8640 W$=MID$(NN$,C,I-C):C=I+1
8650 IF W$="NORTE" OR W$="ESTE" THEN NN$=W$:I=LN
8660 IF W$="SUR" OR W$="OESTE" THEN NN$=W$:I=LN
8670 NEXT I
8680 RETURN
```


Un día un perito fue a un bosque y vio una abeja y un poco de miel. A él le encantaba la miel. dijo me encanta la miel y voy a cogerla, así que trepó y metió la nariz en un agujero. Una abeja lo picó en la nariz. Se fue corriendo a su casa y nunca más volvió a ir al bosque. Fin

Un día un perito fue a un bosque y vio una abeja y un poco de miel A el le encantava la miel. dijo me encanta la miel y voy a cogerla asi que trepó y metió la nariz en un agujero una abeja grande lo pico en la nariz se fue corriendo a su casa y nuunca mas volvio a ir al bosque Fin.

Ayuda didáctica

La utilización de micros puede ayudar eficazmente a desarrollar las aptitudes para la escritura y el dibujo en los niños

Por regla general, las innovaciones técnicas suscitan una respuesta cautelosa por parte del público: la gente suele mostrarse reacia a la experimentación y se niega a apoyar nuevas ideas hasta no haberlas probado por sí mismas. Una de las herramientas básicas de la educación en la actualidad, el cuaderno de ejercicios, representó un desarrollo revolucionario cuando empezó a reemplazar a la pizarra. Previamente, el papel se había venido utilizando mayormente para dejar constancia de transacciones comerciales y para los trabajos importantes de las universidades, el gobierno y la Iglesia. Comodidad cara, se consideró bastante inadecuada para los niños, quienes probablemente no sabrían cómo utilizarlo o lo desaprovecharían.

Sin embargo, la introducción de los cuadernos de ejercicios les proporcionó a los estudiantes un registro permanente de su trabajo. Les permitió dividir su atención entre diferentes tareas escritas sin tener para ello que borrar sus esfuerzos anteriores. A pesar de que fue una innovación cara, fue inmensamente valiosa. No obstante, por cuanto concierne a los argumentos de la economía, el empleo de los micros despierta objeciones similares a las que generaron en su día los cuadernos de ejercicios. El concepto de un ordenador por niño es inaceptable para muchas personas sólo desde el punto de vista económico y, lamentablemente, éste ha sido siempre un factor preponderante para la determinación de la política educativa.

Por cuanto respecta a los micros existe, asimismo, la dificultad de persuadir a los maestros de que acepten una tecnología nueva que ya ha adquirido una reputación dudosa, principalmente a consecuencia de la herencia de las técnicas de la "enseñanza programada" que vimos en el capítulo anterior. Una mirada atenta a la forma en que los ordenadores se pueden utilizar en la clase despejará muchas objeciones y demostrará que el ordenador no sólo aumenta la eficacia (resultando más económico de lo que sugeriría su costo inicial), sino que también puede suponer una valiosa contribución a los procesos creativos del aprendizaje.

Escritura creativa

Consideremos primero el uso de los micros para ayudar a desarrollar aptitudes para la escritura. Escribir una historia implica varios procesos creativos diferentes. Tras la conceptualización, antes de completar la historia es necesario reescribirla, editarla y mejorarla. En este sentido, se considera que escribir es muy similar a, supongamos, pintar y esculpir.

Lamentablemente, la forma en que se espera que muchos niños produzcan trabajo escrito en las

Juego de palabras
Uno de los problemas principales con los que se encuentra el niño cuando intenta escribir de forma creativa es la incapacidad de sus aptitudes de escritura manual para marchar al mismo paso que sus pensamientos. En el proceso de formar las letras en la forma correcta, es muy fácil perder el hilo del pensamiento. Por este motivo es característico que la redacción del niño parezca desarticulada. El procesador de textos no soluciona este problema; hallar una tecla puede ocupar tanto tiempo como escribir la misma letra a mano. Pero con unas adecuadas aptitudes de mecanografía, que exigen menos aptitudes motoras de precisión, el niño puede en cierta medida superar esta dificultad. Puesto que se puede efectuar la edición ya sea en pantalla o en papel antes de contar con un producto ya acabado, se puede invertir menos tiempo en hacer que la historia aparezca en el papel, por lo cual los pensamientos y las palabras se articulan en un grado superior

Micro-cognición
El empleo de un ordenador para tratamiento de textos y dibujo desarrolla muchas de las mismas aptitudes que la escritura y el dibujo manuales. Pero el ordenador, utilizado adecuadamente, en realidad puede enseñar o reafirmar muchas aptitudes mejor que si se las aprendiera a mano, y desarrolla asimismo otras adicionales. En las tablas se relacionan algunas de las aptitudes cognoscitivas que los ordenadores pueden enseñar o perfeccionar

Aptitudes cognoscitivas del tratamiento de textos

- Aptitudes motoras precisas (pequeños movimientos de dedos y manos)
- Reconocimiento de letras
- Formación de letras (un juego de caracteres apropiado en la pantalla y la impresora pueden enseñar la forma correcta de dibujar una letra)
- Corrección de pruebas y revisión de errores
- Ortografía (múltiples aptitudes específicas)
- Secuenciación (colocación de los eventos en el orden correcto)
- Memoria visual (aprender la posición de las teclas de modo que no sea necesario mirarlas)

Aptitudes de los gráficos

- Coordinación ojo-mano
- Aptitudes motoras de precisión
- Diseño
- Discriminación de formas
- Relación causa-efecto
- Fluidez (generación de muchísimas ideas con mucha rapidez)
- Flexibilidad (pensamiento lateral: ver lo antiguo bajo una nueva luz)
- Imaginación

escuelas se opone al proceso creativo. Con frecuencia, la exigencia primordial del maestro es que acaben el trabajo lo más rápidamente posible. También han de presentarlo pulcro, sin tachaduras ni correcciones. El deseo del maestro de obtener un trabajo limpio suele ser perjudicial para la edición, la nueva redacción y, por tanto, la escritura creativa seria. La actitud del maestro está determinada en gran medida por la necesidad de procesar el trabajo escrito con la mayor rapidez posible, y corregir los trabajos sucios lleva más tiempo.

Permitir que un niño utilice un ordenador como procesador de textos reconcilia el síndrome "de página pulcra" con las aptitudes para la escritura seria. Los niños aprenden a editar, corregir y mejorar su trabajo y aun así terminan con una copia final pulcramente impresa. El texto se puede almacenar en disco y cuando la tarea está acabada el niño puede entregarle el disco al maestro, quien lo podrá colocar en un ordenador en su casa o en la escuela y revisarlo cuando sea conveniente. Si se utiliza una red de ordenadores, el maestro puede llamar directamente al archivo del niño y controlar su trabajo. El producto final es una copia pulcra sobre papel de la que hasta el escritor más descuidado se sentirá orgulloso. Existen, asimismo, programas que revisan un archivo de texto para comprobar la ortografía, la puntuación y corregir cualquier eventual error.

Un posible inconveniente de este proceso es que el niño, al ver el texto corregido, no adquiriera conciencia de sus errores como ocurriría en el caso de un cuaderno de ejercicios corregido. Muchas personas que utilizan procesadores de textos prefieren corregir su manuscritos sobre el papel en vez de en la pantalla. Es más fácil para los ojos, el papel es más portátil que el ordenador, y se pueden garabatear comentarios sobre el texto. A menos que se desarrolle un ordenador que sea tan amable con el usuario como un trozo de papel para corregir el trabajo escrito, es probable que el lápiz y el papel permanezcan con nosotros aún durante muchos años.

El hecho de que un niño aprenda a utilizar un ordenador ocasiona muchos comentarios sobre la "barrera del teclado", pero se habla muy poco de la "barrera del lápiz" cuando se aprende a escribir a mano. Uno de los principales objetivos de los maestros de preescolar o de primaria es el de enseñar al niño a escribir. Como primer paso éste debe aprender a sostener el lápiz correctamente, luego aprender las formas de las letras y, por último, reproducir esas formas sobre el papel. Las letras se deben dibujar de una manera determinada y las partes de las letras en un orden determinado. El niño debe hacer palabras de un tamaño uniforme y asegurarse de trazarlas a lo largo de una línea recta. Dominar estas destrezas requiere años de arduo trabajo y muchos niños jamás consiguen escribir adecuadamente. Como si esto no fuera suficiente, el niño de ocho o nueve años debe volver a aprender a escribir con letras cursivas o "unidas".

Compare este proceso con aprender mecanografía al tacto. Uno debe aprender a reconocer las diferentes formas de letras y sus posiciones en el teclado, y a alcanzar ciertas letras con dedos específicos. Esto es mucho más fácil, aunque algunas personas podrían aducir que el desafío que supone aprender a escribir es de gran valor y no se puede ignorar. Pero dominar la escritura a mano a menudo es una barrera para la expresión escrita. Si a un grupo de niños se les asignara cierto límite de tiempo para producir un trabajo, la producción de cada

Diferencias artísticas
Estos tres dibujos los realizó Timothy Ginn, de ocho años de edad. Todos comparten varias características comunes: gran atención al detalle, sombreado para realzar la separación entre objetos y exactitud en la colocación y proporción de éstos, por ejemplo. Sin embargo, tras poco tiempo de aprendizaje con el ratón y el *Macpaint*, Timothy creó un dibujo en el Apple Macintosh que supone varios avances significativos: perspectiva, dado que la nave aparece en tres dimensiones; un fondo, que falta en los otros dos dibujos hechos a mano alzada, y movimiento, ilustrado por la huella de partículas que va dejando la nave. Diferencias como éstas pueden ser aun mayores en niños con trastornos de aprendizaje: con frecuencia pueden visualizar una escena, pero son incapaces de dibujar lo que ve su mente

niño dependería de sus aptitudes para escribir a mano. Sólo cabría esperar una línea o dos de un niño de cinco años de edad, tres o cuatro líneas de uno de seis y quizá una página de un niño de siete años. Si se les enseñara a los niños a mecanografiar, los maestros (y los padres) se sorprenderían de la calidad y cantidad de sus resultados.

El procesador de textos, por supuesto, no ha hecho que la escritura manual quede obsoleta, pero en el transcurso de la vida la práctica común dicta cada vez más la utilización de un teclado, tanto para los negocios como para las situaciones sociales. La habilidad con el teclado a menudo sólo se les enseña a los jóvenes en cursos comerciales y la extensión de estas aptitudes a los niños en edad escolar es beneficiosa. Esta tendencia continuará en la medida en que la presencia de ordenadores se vuelva más familiar y su empleo más diversificado.

Trabajo de diseño

Otro aspecto de la microinformática que reviste gran importancia para el niño es la creciente facilidad con que se pueden crear sofisticados gráficos gracias a la utilización de un ordenador. Este papel del micro se ha visto seriamente limitado por su precio: el proceso de gráficos por lo general requiere un hardware más complejo y más caro que el proceso de textos. Sin embargo, esta característica de la informática escolar está evolucionando, puesto que simultáneamente las máquinas se están volviendo más potentes y menos costosas.

Al principio, la tablilla para gráficos permitía utilizar en la pantalla varias funciones artísticas, pero recientemente estas características se han incorporado en forma de "tecnología de ratón", entre otras máquinas, en el Apple Macintosh. Al permitir la selección de "iconos", el ratón le ofrece al usuario un control preciso e inmediato sobre una amplia gama de actividades, desde dibujar imágenes y formas a reducir y ampliar imágenes, rellenando superficies con patrones y manipulando la visualización de diversas maneras.

Además, el ordenador le permite al estudiante integrar y relacionar material diferente, textual y gráfico, con más facilidad que los métodos tradicionales. Una imagen acabada se puede "recortar" y "pegar" en un fragmento de texto generado anteriormente. Aunque los programas para gráficos tales como el *Macpaint* no pueden sustituir a las prácticas artísticas que se realizan en la clase, sí le ofrecen al niño un medio con nuevas técnicas con las que experimentar, aprender y expresarse.

El niño desea aprender. Su curiosidad innata lo obliga a pasarse cada una de las horas de vigilia buscando, explorando y aprendiendo todo lo que puede acerca de sus nuevos entornos, y expresando sus impresiones a través de un impulso creador. Los lápices de colores se cogen con ansiedad, dibujando sobre el papel o la pared más cercanos: así se van desarrollando estas aptitudes básicas.

Los niños de dos años con acceso a un ordenador efectúan sus investigaciones con el mismo regocijo, pulsando todas las teclas y viendo aparecer en la pantalla pequeñas formas. Aceptan la nueva tecnología con mayor facilidad que los adultos y, al encontrarse con un entorno que contiene ordenadores, el niño los utiliza más fácil y naturalmente. Ahora ya sólo es cuestión de tiempo para que las escuelas incorporen el micro como una herramienta educativa vital y dejen de considerarlo sólo como un juguete muy caro.

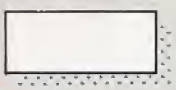
Tipos de variables

En este capítulo estableceremos las diferencias entre las variables y estudiaremos las sentencias compuestas

Símbolos del PASCAL
Estas tres formas son los símbolos que se suelen usar en los diagramas sintácticos:

- El símbolo redondeado representa las palabras reservadas en PASCAL o los caracteres que no necesitan otra explicación (como 'letra' o 'dígito')

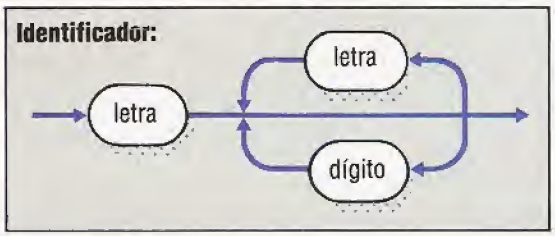
- Un círculo representa un operador en PASCAL (+, −, *, ., etc.)



- El símbolo rectangular representa una palabra o frase que posee su propio diagrama de sintaxis separado

El PASCAL proporciona cuatro tipos de datos simples ya predefinidos para nosotros, y a los que se les asignan los identificadores Integer (enteros), Real (reales), Char (caracteres) y Boolean (booleanos). Los números se clasifican según posean una parte de fracción (números reales) o sean números enteros naturales (enteros). Por supuesto, la verdadera gama de números disponibles está determinada por la cantidad de bytes que se empleen para almacenar un valor dado de cada tipo. Las decisiones de diseño como ésta las toma el escritor del compilador (el *implementador*) y se dice que tales características están *subordinadas a la implementación*. Todas estas subordinaciones a la implementación deben especificarse en la documentación para que un compilador sea autorizado por la normativa de la ISO (Internacional Standards Organization).

Es casi seguro que la gama de enteros de su compilador esté comprendida entre −32.768 y 32.767, o bien entre −2.147.483.648 y 2.147.483.647, según se utilice una representación de dos o cuatro bytes. El PASCAL posee un nombre para el valor del entero



máximo, representado mediante el identificador de constante predefinida, MaxInt. Por tanto, puede hallar fácilmente este valor mediante la sentencia:

```
WriteLn ('MaxInt is :',MaxInt)
```

Los números reales también se mantienen en una gama y exactitud limitadas: por lo general desde alrededor de 1.7E+38, con una precisión de seis o siete dígitos en el peor de los casos. Esta forma de escribir los números reales, llamada *notación científica*, es la forma por defecto del PASCAL, pero se puede utilizar la manera más normal de escribirlos, con el punto decimal (p. ej., 123.456), si así lo cree conveniente.

Se ha de tener en cuenta, no obstante, que un número real *siempre* posee un punto decimal separando la parte entera de la parte fraccionaria, y que *ambas deben* estar presentes. De modo que 0.1 y 1.0E-1 son aceptables, pero .1 o 1E-1 son ilegales. Afortunadamente, estas reglas estrictas sólo se aplican a números que en el texto del programa se deban reconocer como reales. Si usted está entrando datos desde el teclado, por ejemplo, un entero se leerá y se convertirá automáticamente si se espera un valor real.

Char es la abreviatura de carácter, por supuesto, y un valor de este tipo será uno de los miembros del juego de caracteres (por lo general ASCII) de que disponga el ordenador. El PASCAL asegura su propia portabilidad dando por sentado que:

- Los caracteres de la A a la Z están ordenados alfabéticamente, lo que significa que, según el valor de los caracteres, A es menor que B, B menor que C, y así sucesivamente.
- Los caracteres de números del 0 al 9 están ordenados y contiguos, lo que significa que, sea cual fuere el valor de 0, 1 será el siguiente, etc.

El juego de caracteres ASCII también posee un alfabeto contiguo, pero ello no es esencial para el PASCAL, ¡sino para los programadores de este lenguaje! Cada valor de carácter tendrá un código numérico, que es un valor de un subrango del tipo de enteros. Los códigos ASCII están definidos en la gama entre 0 y 127, y muchas máquinas lo amplían hasta 255 para códigos extras de caracteres para gráficos. Podemos fácilmente hacer un mapa de cualquier juego de caracteres en la escala de los valores "comunes" que utiliza internamente el ordenador. El PASCAL proporciona la función predefinida Ord: ésta devuelve el código de entero de su argumento; de modo que Ord (A) es el equivalente de 65 en el juego de caracteres ASCII. Otra función, chr, proporciona la función inversa: chr (65) da el carácter A (observe que con chr *no* se utiliza el signo dólar).

Tanto el rango de los valores de caracteres como los enteros se definen para cada implementación, y existen en una escala ordenada de constantes conocidas. Por este motivo se los denomina tipos *ordinales* o *escalares*. Cualquiera que sea el valor, siempre sabemos cuáles son los anteriores y los siguientes, si es que los hay. Estos valores adyacentes se pueden obtener mediante las dos funciones escalares:

```
pred(elemento) (predecesor)
succ(elemento) (sucesor)
```

Por consiguiente, succ (3) dará siempre el valor de carácter 4, pero pred(Z) sólo será Y en algunos juegos de caracteres, como el ASCII. Pred (MaxInt) será ya sea 32.766 o bien 2.147.483.646. La función chr sólo se puede utilizar con un argumento que sea un código de carácter. Todas las otras funciones escalares se pueden emplear con cualquier tipo escalar, aunque si se utiliza ord con enteros, devuelve el valor de su argumento.

Las variables booleanas son el más simple de todos los tipos escalares, porque en la escala hay sólo dos valores: false (falso) y true (verdadero), por ese orden. Puesto que son tipos escalares simples, las funciones escalares se pueden aplicar a cualquier valor booleano: el valor ordinal de false es 0 y ord (true) es 1. Las otras funciones escalares, pred y succ, sin embargo, no son de gran utilidad aquí. Su

compilador de PASCAL desaprobará categóricamente que usted pruebe algo como WriteLn(pred(false)): no debe sorprenderle que sea un error intentar evaluar un valor inexistente.

La siguiente parte de definición de constantes de un programa muestra todos los tipos ordinales simples tal como podrían aparecer en un texto fuente.

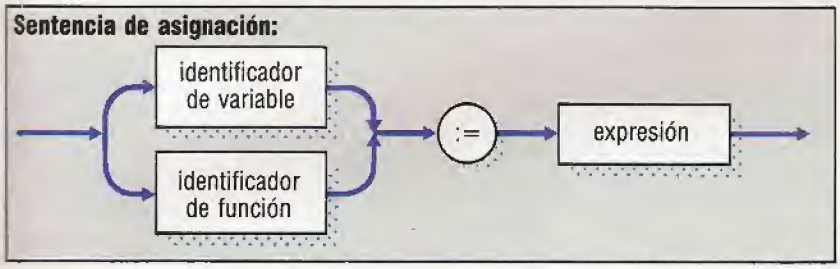
```
CONST
  VAT          = 0.15;
  columnas     = 40;
  espacio      = ' ';
  depuracion   = false;
```

Igualdad y asignación

El signo de igualdad (=) *siempre* significa es igual a en PASCAL, y se utiliza para igualar identificadores de constantes a los valores que retienen. Cuando declaramos variables en la sección de declaraciones VAR, los dos puntos (:) separan al identificador de variables recién definido de su tipo. Por ejemplo:

```
VAR
  coeficiente    : real;
  numero         : integer;
  simbolo        : char;
  hecho          : boolean;
```

Cuando queremos asignar valores a estas variables, se utiliza el *operador de asignación* compuesto (:=). Éste ayuda a diferenciar claramente las tres clases de operación. Las definiciones de CONST igualan valores permanentes, las declaraciones VAR sólo reservan espacio de memoria, y asignación le da al identificador un valor (tal vez temporal).



Liz Dixon

La sentencia compuesta

Cuando se deben ejecutar dos o más sentencias como parte de un único proceso, podemos encerrarlas entre paréntesis entre las palabras BEGIN y END como una sentencia "compuesta". Recuerde que cada sentencia integrante debe estar separada de cualquier sentencia que le siga mediante un punto y coma. Ya hemos visto sentencias compuestas, dado que el cuerpo de sentencias ejecutables de todos los programas asume esta forma. A continuación ofrecemos un programa completo que utiliza muchas de las características que ya hemos considerado. Adoptaremos la convención de escribir las palabras reservadas en mayúsculas para distinguirlas de los identificadores. Observe las líneas dejadas en blanco entre cada parte de la estructura del programa, para facilitar su lectura:

```
PROGRAM Circulo(input,output);

CONST
  pi             = 3.1415926536;
  aviso          = 'Entre el radio:';

VAR
  radio,
  superficie     : real;

BEGIN
  WriteLn;
  write(aviso);
  read(radio);
  superficie:=pi*radio*radio;
  WriteLn;
  WriteLn('La superficie de un circulo',
    'de radio',radio : 8 : 3);
  WriteLn('es:',superficie : 10 : 3)
END.
```

En este ejemplo hay dos aspectos sintácticos que hemos de observar. Primero, la parte VAR declara dos identificadores del mismo tipo, ambos reales. No se necesitan, sin embargo, dos declaraciones separadas, dado que las listas de elementos simplemente se separan mediante una coma; esto es universal en PASCAL. Por consiguiente, cuando especificamos más de un argumento para un procedimiento (como en las sentencias WriteLn) se aplica la misma sintaxis. La segunda característica nueva es el formato de salida que se utiliza para evitar la notación científica por defecto de los valores reales. Opcionalmente, podríamos especificar dos enteros, separados por dos puntos, para forzar una cierta anchura de campo para el número entero y su parte fraccional.

En nuestro programa Circulo, tanto al radio como a la superficie se le concederán tres posiciones decimales. Puesto que la superficie será un número mayor (y, por tanto, más largo), le concedemos un total de 10 posiciones en vez de las ocho que le concedemos al radio. Estos valores enteros deben ser mayores que cero, permitir un posible signo, tener al menos un dígito y acomodar el punto decimal, escrito antes de la parte fraccionaria. Los valores ilegales producirían errores en el momento de la ejecución, o (en el mejor de los casos) harían que el formato saliera en notación científica; WriteLn (X : 6 : 2) no dejaría sitio para números mayores que 99.99, por ejemplo.

Más valioso aún es el hecho de que el PASCAL redondeara automáticamente el último dígito para dar la mayor exactitud en cualquier campo numérico solicitado. Además, se puede utilizar cualquier variable o expresión y no tan sólo una constante. Ello permite una enorme flexibilidad, incluyendo facilidades para tabulación. Con todos los otros tipos de datos, sólo se necesita un valor entero (en realidad, sólo se permite uno) para especificar la anchura del campo. Especificar una anchura de uno haría que los enteros se escribieran en el campo de dimensiones mínimas, sin espacios. Por lo tanto, hemos de recordar colocarlos nosotros mismos si es que han de tabularse los resultados. Por ejemplo:

```
WriteLn('Total:' : 20, peso : 1,'toneladas.')
```

Normalmente el PASCAL se negará a dar una salida confusa o inexacta, pero la capacidad para suprimir todos los espacios implica que debemos tener cuidado de no imprimir dos números consecutivos con un campo de uno. Por ejemplo, si 12 y 34 se escribieran de este modo, la salida sería 1234.

Hacia el Nuevo Mundo

Presentamos el proyecto de un juego de simulación basado en un viaje a las Indias durante el s. XVI

Además de proporcionar a los entusiastas de los "marcianitos" juegos de acción rápida que ponen a prueba los reflejos, el ordenador es un medio excelente para un tipo de desafío bastante diferente: el juego de simulación. Éste implica aptitudes de naturaleza muy diferente a las que se requieren para los juegos recreativos. Tal como lo indica la palabra "simulación", al objeto del juego se crea un modelo basado en ordenador del mundo real. Estos modelos pueden ser simuladores en "tiempo real", como los programas simuladores de vuelo o de estrategia.

En el juego de estrategia, el jugador o el grupo de jugadores tiene asignada una serie de tareas que llevar a cabo y se le proporciona información para ayudarlo a tomar las decisiones. Un ejemplo típico es un juego mercantil, en el cual a los participantes se les entregan sumas de dinero y deben comerciar con mercancías con el fin de obtener los máximos beneficios. Lo que hace que esta clase de simulación sea diferente de, pongamos por caso, un juego de aventuras, es el hecho de que mientras el jugador tiende a tropezar con obstáculos y peligros a medida que se va desplazando por el mundo de aventuras y los va sorteando a medida que se le van presentando, en un juego de simulación el jugador a menudo va planificando por adelantado, administrando sus recursos para hacer frente a las contingencias esperadas (y también a las imprevistas).

Mundos de ficción
En la fotografía vemos ejemplos de los tres tipos principales de simulación. *Space shuttle* es una realista simulación estilo recreativo de la lanzadera espacial auténtica. *Air traffic control* es una simulación en tiempo real en la cual el jugador ha de impartir instrucciones a varios aviones para que puedan despegar y aterrizar con seguridad. *The great nordic war* es una simulación de la guerra de los Treinta años, asumiendo el jugador el papel del rey de Suecia

Este tipo de juego exige a los participantes, por lo tanto, aptitudes de administración y previsión; en otras palabras, capacidad para planificar con antelación en vez de "sobre la marcha". Por supuesto, para reflejar el mundo real en un juego de simulación no se puede permitir que todo salga exactamente tal como se planeó; una buena simulación siempre introducirá factores aleatorios que alteren los esquemas del jugador. Un buen jugador, por consiguiente, se debe ocupar a fondo en la planificación de contingencias, para minimizar el daño que lleguen a infligir los sucesos imprevisibles.

Los juegos de simulación poseen la ventaja de permitir que las personas participen formando un equipo, combinando su talento y sabiduría en contra del ordenador, para alcanzar el objetivo prefijado. Por este motivo los juegos de simulación se están comenzando a utilizar en las escuelas, dado que permiten que el niño modele la clase de comportamiento que cabría esperar de él en el mundo de trabajo de los adultos, combinando los recursos disponibles y los talentos de un pequeño grupo para alcanzar cierto objetivo. Participar en juegos de simulación es, asimismo, sumamente entretenido.

Además de todas estas cualidades, los juegos de simulación cubren un área que muy raramente abordan la mayoría de las aplicaciones para ordenador: se pueden suscitar cuestiones morales, teniendo los jugadores que sopesar, por ejemplo, el aumento de la rentabilidad contra el bienestar de los empleados. Cuando tales decisiones se toman en equipo, suelen ir precedidas de acalorados debates acerca de los pros y los contras.

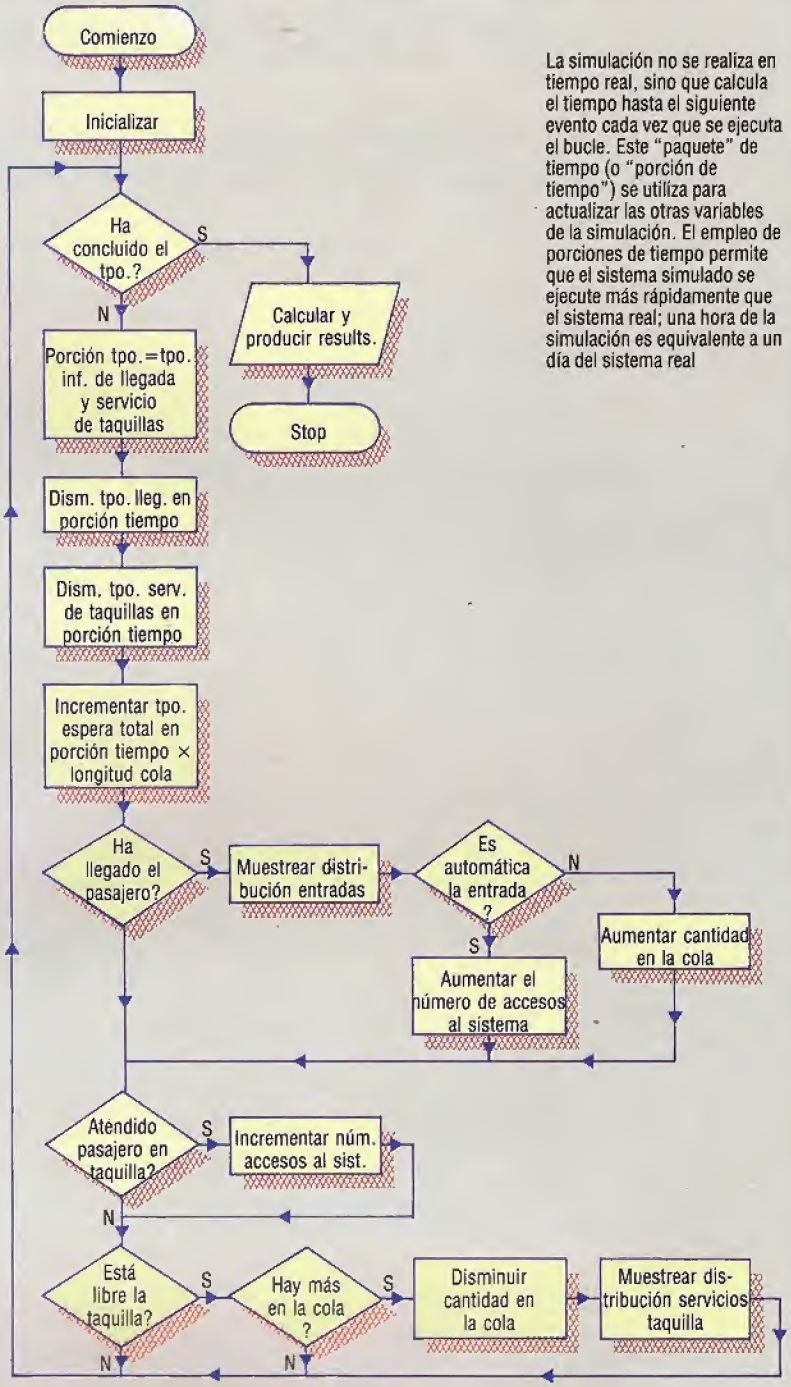
Existen muchos escenarios diferentes que se prestan bien para la simulación por ordenador; a menudo el factor común es que la simulación implica llevar a cabo una tarea específica para alcanzar un objetivo muy concreto. En algunos casos éstos los puede seleccionar el jugador. El objetivo podría ser permanecer en el negocio durante un tiempo dado o, por el contrario, ganar millones de pesetas. Muchas simulaciones se basan en la estrategia financiera, pero algunas poseen objetivos más filantrópicos. Un juego de simulación que se utiliza actualmente en las escuelas primarias de Gran Bretaña es un proyecto para localizar y sacar a la superficie los restos del naufragio del galeón isabelino *Mary Rose*. En este juego a los participantes también se les entregan documentos para complementar el programa y proporcionar pistas adicionales que ayuden al proceso de tomar decisiones.

En esta serie de proyectos de programación construiremos un juego de simulación mercantil, situando a los jugadores en condiciones de tener que equipar y navegar un barco hasta el Nuevo Mundo, en el s. XVI. El juego está construido como una serie de módulos que se encajan entre sí, y el pro-

Bajo tierra

En la actualidad el uso de las simulaciones está muy generalizado en el campo de la educación y de los juegos, pero las primeras simulaciones basadas en ordenador se desarrollaron como una herramienta de diseño y gestión. Las simulaciones revisten gran valor para el diseño del flujo de tráfico y sistemas de servicios, en los cuales la naturaleza del modelo los hace ideales para la simulación por ordenador. Tales simulaciones por lo general se basan en principios estadísticos y matemáticos que predicen distribuciones de frecuencia de las variables utilizadas.

Podemos demostrar los principios de una simulación analizando un simple sistema de servicios. El sistema existente permite el acceso a la estación de metro mediante la compra de un billete en una taquilla. Se propone instalar una cantidad de barreras de billetes automáticas que operan con billetes adquiridos con anterioridad. El diagrama de flujo ilustra los procesos implicados en la creación de una simulación sencilla del sistema. Se utilizan tres distribuciones: una de llegada de pasajeros, otra de horario de funcionamiento de la taquilla, y una tercera para el método de acceso (es decir, manual o automático). Las dos primeras distribuciones provienen de observaciones estadísticas y métodos matemáticos; la tercera variará a cada ejecución del programa con el fin de observar las ramificaciones de diversas proporciones de puertas de acceso manuales y automáticas. Esta simulación calcula el tiempo de espera promedio de los pasajeros



La simulación no se realiza en tiempo real, sino que calcula el tiempo hasta el siguiente evento cada vez que se ejecuta el bucle. Este "paquete" de tiempo (o "porción de tiempo") se utiliza para actualizar las otras variables de la simulación. El empleo de porciones de tiempo permite que el sistema simulado se ejecute más rápidamente que el sistema real; una hora de la simulación es equivalente a un día del sistema real

grama está diseñado de modo tal que el escenario se pueda modificar con facilidad, proporcionando un esqueleto alrededor del cual se podría construir un contexto diferente. Por lo tanto, la simulación se podría modernizar para convertirse en un viaje de ciencia-ficción, comerciando con un planeta lejano a través de una nave espacial.

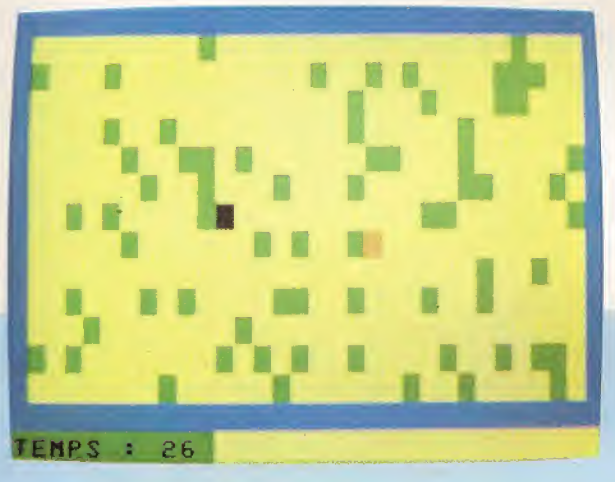
El objetivo del juego, para uno o más jugadores, consiste en comerciar con las mercancías obteniendo los máximos beneficios, pero equilibrando esta finalidad con la seguridad y el bienestar de la tripulación. El juego se dividirá en tres etapas: preparación, viaje y comercio. La primera fase supone reunir los recursos y planificar el viaje, y la segunda introduce en el juego elementos al azar. Su habilidad para hacer frente a estas contingencias dependerá en gran medida de lo bien que haya elaborado su plan durante la primera fase del juego. La tercera fase implica establecer buenas relaciones con los nativos y comerciar con ellos de forma rentable.

Desde el punto de vista del programador, la construcción del juego se basará en una serie de módulos independientes y ofreceremos complementos al BASIC para el BBC Micro, el Spectrum y el Commodore 64. Las principales tareas del programa son llevar el registro de las decisiones tomadas por el jugador, crear diversas contingencias y analizar y controlar el estado del jugador a medida que va avanzando el juego. En el próximo capítulo abordaremos el primer módulo del programa.

Persecución

Pongamos a prueba su habilidad para atrapar a un amigo de lo ajeno. El programa en BASIC está escrito para el microordenador Alice

El ladrón ha escapado llevándose el botín. Se esconde en la ciudad, y usted tiene treinta minutos para encontrarlo y detenerlo. Atención, ¡no se precipite! En efecto, si se echa sobre él sin pensar, el fugitivo tiene todas las posibilidades de escapársele.



La mejor manera de atraparlo es alcanzándolo de lado. (Es el método más efectivo a condición de no fallar.) Si no se siente lo bastante seguro de sí mismo, atáquelo de frente, lo cual es más fácil pero mucho menos eficaz, ya que no es tan discreto. Otro consejo: no intente perseguirlo; no daría resultado, pues él es mucho más rápido que usted. Ha de observar sus movimientos como si fuera un detective. Cuando vea que da la vuelta, acérquese sigilosamente y sorpréndalo en el momento justo. Pero recuerde: ¡el tiempo va pasando!

Desplazamiento:
Z =arriba
Q =izquierda
S =derecha
W =abajo

```
5 REM ********************************
10 REM *          PERSECUCION         *
15 REM ********************************
19 REM S=PUNTOS
20 S=0
25 REM V$=CARACTER LADRON
30 V$=CHR$(128)
35 REM P$=CARACTER JUGADOR
40 P$=CHR$(191)
50 GOSUB 2000
104 REM
105 REM BUCLE PRINCIPAL
106 REM
109 REM MOVIMIENTO JUGADOR
110 D$=INKEY$
120 D=(D$="Q")-(D$="S")+32*((D$="Z")-(D$="W"))
130 IF D<>0 THEN D0=D
135 REM DESCONTAR TIEMPO
150 T=T-0.1
160 PRINT@ 480,"TIEMPO :";INT(T+1);
165 REM TIEMPO CONSUMIDO?
170 IF T<0 THEN 490
180 P=P+D0
190 C=PEEK(16384+P)
195 REM LADRON ATRAPADO?
200 IF C=128 THEN 4000
205 REM OBSTACULO?
210 IF C<>159 THEN P=P1
220 PRINT@ P1,CHR$(159);
230 PRINT@ P,P$;
240 P1=P
245 REM DESPLAZAMIENTO LADRON
250 V=V+DV
255 REM OBSTACULO?
260 IF PEEK(16384+V)<>159 THEN GOSUB 600
270 IF PEEK(16384+V)<>159 THEN 250
280 PRINT@ V1,CHR$(159);
290 PRINT@, V,V$;
300 V1=V
310 GOTO 110
484 REM
485 REM FIN
486 REM
490 D$=INKEY$
500 IF R<S THEN R=S
510 PRINT@ 166,"TIEMPO CONSUMIDO";
520 PRINT@ 234,"PUNTOS :";S;
530 PRINT@ 266,"RECORD :";R;
540 PRINT@ 326,"OTRA ?";
550 D$=INKEY$
560 IF D$="" THEN 550
570 IF D$<>"N" THEN 20
580 END
594 REM
595 REM OBSTACULO
596 REM
600 D2=D2+1
610 GOSUB 900
620 IF PEEK(16384+V1+DV)=159 THEN
    V=V1+DV:RETURN
630 D2=D2-2
640 GOSUB 900
650 IF PEEK(16384+V1+DV)=159 THEN
    V=V1+DV:RETURN
660 D2=D2-1
670 GOSUB 900
680 V=V1+DV
690 RETURN
900 IF D2>4 THEN D2=D2-4
910 IF D2<1 THEN D2=D2+4
920 DV=(D2=1)-(D2=3)+32*((D2=2)-(D2=4))
930 RETURN
1994 REM
1995 REM INICIALIZACION
1996 REM
2000 CLS 2
2005 REM TRAZAR CUADRADO
2020 FOR I=0 TO 31
2030 PRIT@ I,CHR$(175);
2040 PRINT@ 448+I,CHR$(175);
2050 NEXT I
2060 FOR I=1 TO 13
2070 PRINT@ I*32,CHR$(175);
2080 PRINT@ I*32+31,CHR$(175);
2090 NEXT I
2095 REM INSTALACION OBSTACULOS
2100 FOR I=1 TO 70
2110 GOSUB 3000
2130 PRINT@ P," ";
2140 NEXT I
2145 REM CUADRO LADRON
2150 GOSUB 3000
2160 V=P
2170 PRINT@ V,V$;
2180 V1=V
2185 REM CUADRO JUGADOR
2200 GOSUB 3000
2210 PRINT@ P,P$;
2220 P1=P
2230 T=30
2240 D0=0
2250 DV=0
2260 D2=0
2270 RETURN
2994 REM
2995 REM POSICION ALEATORIA
2996 REM
3000 P=RND(414)+32
3005 REM POSICION OCUPADA?
3010 IF PEEK(16384+P)<>159 THEN 3000
3020 RETURN
3994 REM
3995 REM GANA
3996 REM
4000 FOR I=1 TO 5
4005 REM SIRENA
4010 SOUND 35,10
4020 SOUND 5,10
4030 NEXT I
4040 S=S+1
4050 GOTO 50
```

Ritmo y melodía

El CX5M aprovecha la experiencia de Yamaha en sonido y se convierte en el primer ordenador personal exclusivo para música

Aunque adherido al estándar MSX, que permite su utilización para aplicaciones tales como juegos o proceso de textos, el ordenador Yamaha CX5M está decididamente dirigido al amante de la música electrónica. Se puede conectar un teclado de piano externo YK-10 o YK-01 en un conector situado a uno de los lados de la máquina, donde hay asimismo un par de puertas MIDI (interface digital para instrumentos musicales) que posibilitarán la ejecución desde el ordenador de cualquier sintetizador o dispositivo para música electrónica con una interface MIDI. Hay igualmente un par de conectores microjack para el empleo de altavoces externos.

El ordenador tiene un aspecto muy parecido al de las otras máquinas de la gama MSX. Hay 48 teclas de máquina de escribir rodeadas por 10 teclas de función, que contienen características estándares MSX, tales como un retroceso con borrado, una tecla Graph para visualizar un conjunto de caracteres para gráficos, y una tecla Code, la cual, al ser pulsada junto con una de las teclas de máquina de escribir, imprime toda una variedad de caracteres de idiomas extranjeros. Encima del teclado hay cinco teclas de función, capaces de proporcionar 10 funciones programables. En el encendido, éstas pasan por defecto a las funciones MSX estándares de AUTO, LIST, RUN, etc. En el extremo inferior derecho del teclado está el grupo de las cuatro teclas del cursor y encima del mismo hay un conjunto de cinco teclas que configuran otras instrucciones estándares MSX tales como INSertar, DEL (eliminar) y STOP. Encima del teclado hay una puerta para cartucho.

La arquitectura de la máquina

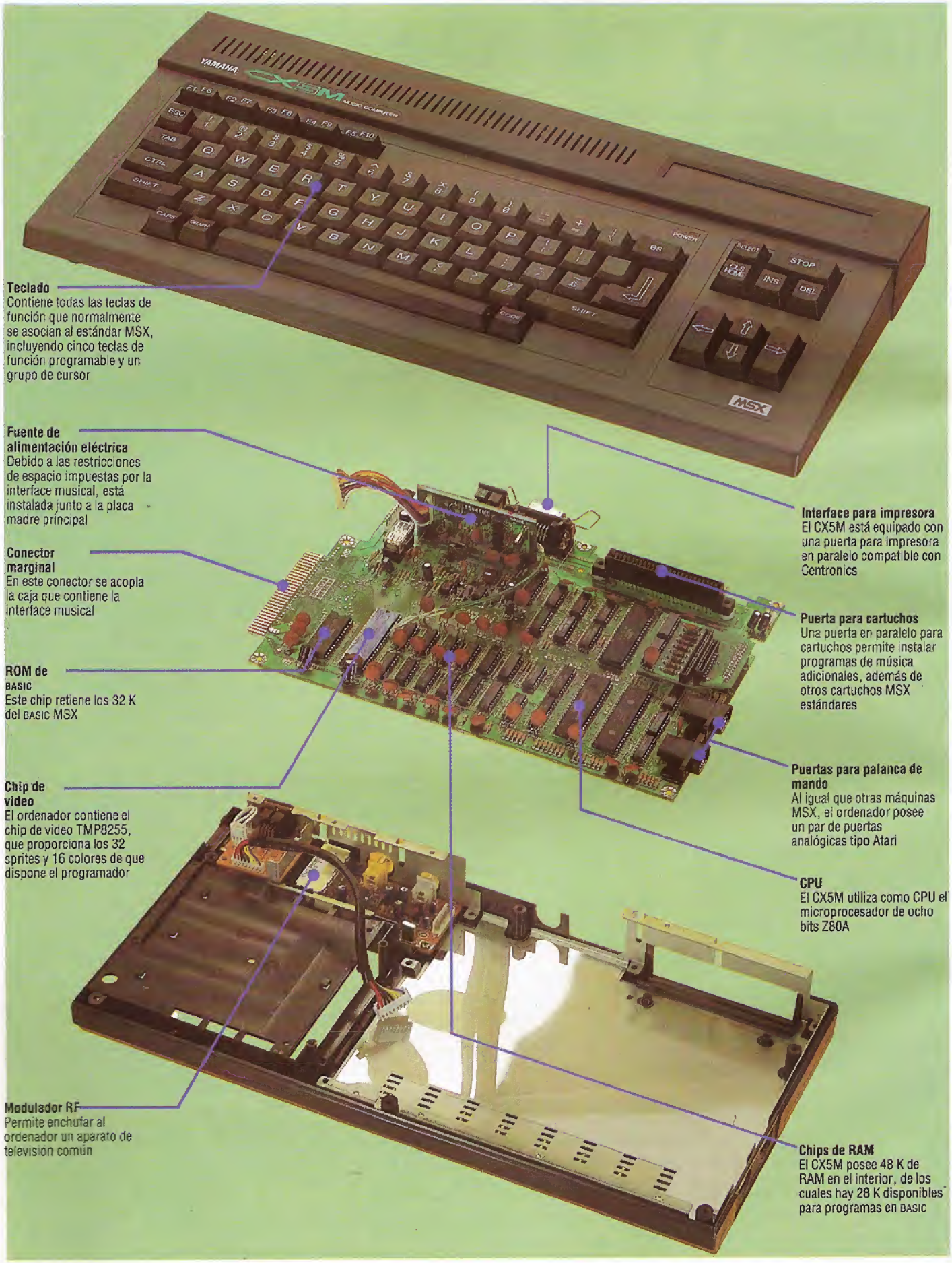
En el lado derecho del CX5M hay un par de puertas para palanca de mando tipo Atari. En su parte posterior, el ordenador posee un conector marginal en paralelo, la interface para impresora compatible con Centronics, una puerta para cassette, un enchufe hembra de sonido para salida mono, un enchufe hembra para monitor de video compuesto, un enchufe para televisión RF y la entrada de la fuente de alimentación eléctrica. Una única caja contiene las puertas MIDI y la interface del teclado, que les otorgan al CX5M sus capacidades musicales exclusivas. Esta caja se introduce en un conector marginal situado debajo de la máquina, al lado izquierdo. La razón de ello reside en que Yamaha pretende producir una serie de interfaces con características diferentes. Cuando estos dispositivos salgan a la venta, los usuarios podrán intercambiar las interfaces simplemente quitando un tornillo y colocando en el conector marginal la interface nueva.

Chris Stevens

El teclado YK-01 de tres octavas y media (que se vende con el ordenador y que podemos apreciar en la fotografía) posee teclas con un agradable tacto profesional, si bien los músicos de teclado serios las encontrarán demasiado pequeñas como para resultar totalmente cómodas. Utilizando el software propio del ordenador, el teclado se puede "dividir", lo que significa que una voz programada se puede tocar en la parte inferior del teclado mientras en el resto de las teclas se puede tocar otra completamente diferente. Esto significa, por ejemplo, que uno puede tocar "cuerdas" en las teclas superiores mientras proporciona con las inferiores un acompañamiento de "contrabajo". El teclado también puede funcionar en modalidad monofónica o polifónica (o ambas), tocando simultáneamente hasta ocho notas.

Tras el encendido, se visualiza la pantalla azul estándar MSX. Se puede acceder al programa de música mediante la instrucción CALL MUSIC, la cual visualizará un menú con cinco bloques de las diversas opciones disponibles. El usuario puede entonces pasar la lista hacia abajo mediante el empleo de la tecla Return, y los parámetros de cada opción se alteran pulsando las teclas del cursor. Los bloques etiquetados POLY y MONO permiten que el músico seleccione cuál de las 46 voces preprogramadas se debe utilizar, y en cuál de las dos modalidades. Estas voces van desde instrumentos musicales convencionales (como órgano, guitarra e instrumentos de percusión como el vibráfono y el cencerro) hasta una gama de sonidos cotidianos (como una "sirena de ambulancia" y "sonido de gotas de lluvia"). Debido a la naturaleza de algunas voces, hay algunas

Cajas de música
Se prevé que el Yamaha CX5M se venderá mejor en las tiendas de instrumentos musicales que en las de informática. Para realzar sus capacidades musicales, el ordenador se comercializa como un paquete junto con un teclado de piano ya sea YK-01 o bien YK-10. El teclado de piano se enchufa en una pequeña caja para interface de música que está atornillada en la parte inferior del ordenador



Teclado
Contiene todas las teclas de función que normalmente se asocian al estándar MSX, incluyendo cinco teclas de función programable y un grupo de cursor

Fuente de alimentación eléctrica
Debido a las restricciones de espacio impuestas por la interface musical, está instalada junto a la placa madre principal

Conector marginal
En este conector se acopla la caja que contiene la interface musical

ROM de BASIC
Este chip retiene los 32 K del BASIC MSX

Chip de video
El ordenador contiene el chip de video TMP8255, que proporciona los 32 sprites y 16 colores de que dispone el programador

Modulador RF
Permite enchufar al ordenador un aparato de televisión común

Interface para impresora
El CX5M está equipado con una puerta para impresora en paralelo compatible con Centronics

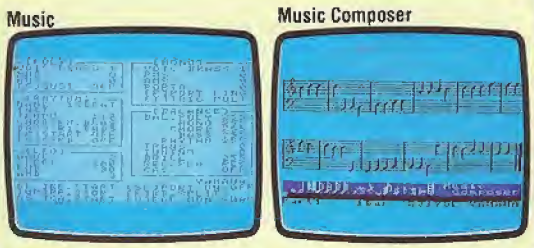
Puerta para cartuchos
Una puerta en paralelo para cartuchos permite instalar programas de música adicionales, además de otros cartuchos MSX estándares

Puertas para palanca de mando
Al igual que otras máquinas MSX, el ordenador posee un par de puertas analógicas tipo Atari

CPU
El CX5M utiliza como CPU el microprocesador de ocho bits Z80A

Chips de RAM
El CX5M posee 48 K de RAM en el interior, de los cuales hay 28 K disponibles para programas en BASIC

Éstos son algunos de los cartuchos que han salido a la venta para el CX5M. Cada uno de los programas que vemos le permite al ordenador llevar a cabo una función diferente, desde programar al sintetizador DX-7 hasta componer programas de música. Cada cartucho viene con su propio manual. Sin embargo, son relativamente caros y los usuarios quizá prefieran esperar hasta que el precio se reduzca o aparezca software perfeccionado



Ian McKinnell

Arriba vemos dos aplicaciones para el CX5M. El paquete Music está incorporado en el ordenador y se accede a él a través de la instrucción CALL MUSIC. Desplazando el cursor con las teclas Function y Return, los parámetros indicados se pueden alterar luego pulsando las teclas del cursor. El Music Composer permite al usuario escribir una partitura musical en la pantalla, que se puede editar y enviar después a la impresora para obtener una salida impresa

que no están disponibles cuando se utiliza el "sostenido". La alteración de las voces de las modalidades POLY y MONO le permite al usuario establecer tonos contrastantes para usar con el teclado dividido: es decir, designar algunas teclas como monofónicas y otras como polifónicas. Es posible, por supuesto, establecer todo el teclado en cualquiera de las modalidades; sin embargo, no se lo puede dividir en dos secciones POLY.

Los sonidos que se pueden producir son notables, aun cuando se emitan a través de un altavoz de televisor normal, si bien algunos de los sonidos guardan poco parecido con los nombres que se les ha dado. Por ejemplo, "tren" suena simplemente como un órgano eléctrico normal. Éste es un defecto común de los fabricantes de sintetizadores; pero, siempre y cuando el usuario no espere una reproducción exacta de los sonidos cotidianos, la mayoría de los músicos se sentirán más que satisfechos con las voces proporcionadas.

Otro bloque, RHYTHM, proporciona un acompañamiento rítmico a una melodía. Son seis los diferentes ritmos disponibles, que se pueden construir utilizando cuerdas, contrabajo y percusión. De éstos, los tonos de contrabajo y cuerdas los puede establecer el músico. El tiempo del ritmo se puede variar y hay asimismo una opción que le permite al músico variar la altura *(pitch)* del ritmo desde el teclado de piano. Alterando la forma de la onda de sonido mediante el bloque LFO (Low Frequency Oscillator), se pueden modular tonos para producir sonidos diferentes. El bloque BALANCE permite el ajuste de volumen independiente de cada elemento rítmico, más las modalides POLY y MONO. Por lo tanto, se puede reducir el acompañamiento rítmico de fondo y elevar el volumen de la modalidad POLY para darle mayor énfasis a la melodía. Cuando el músico se sienta satisfecho con los sonidos producidos, los datos se pueden guardar en cassette y cargar posteriormente. Esto le ahorra al músico el tener que escribir los distintos parámetros que han sido seleccionados y volver a establecerlos después.

Programas para el Yamaha CX5M

Aparte del software incorporado, Yamaha ha lanzado una serie de cartuchos para utilizar con el CX5M. Esta gama de cartuchos incluye el FM Voicing Program, que es una versión ampliada del software incorporado. El programa le permite al músico manipular de una forma más cabal la forma de la voz mediante el ajuste de las frecuencias y los algoritmos con los que fue construida. Un segundo programa de voz en cartucho le permite al ordenador programar el sintetizador Yamaha DX7. El FM Music Macro permite programar el ordenador con un conjunto de instrucciones adicionales en BASIC, mientras que el FM Music Composer visualiza un pentagrama en blanco y permite entrar una partitura convencional y volcarla luego a una impresora externa.

Aunque podría parecer que Yamaha ha proporcionado un amplio juego de programas para el CX5M, uno se queda con la sensación de que el software no explota totalmente las capacidades de la máquina. Los programas de modulación, en especial, parecen un poco limitados; en ocasiones resulta difícil discernir cualquier cambio perceptible en el sonido de una nota. Esto es un hecho inusual, puesto que Yamaha ha basado su sistema operativo en el utilizado con el sintetizador DX7, cuya enorme gama de sonidos hace que, en comparación, el CX5M parezca insignificante. Además, aunque los métodos elegidos por Yamaha para alterar los parámetros de los bloques sean adecuados, en ocasiones parecen engorrosos; por ejemplo, utilizar las teclas del cursor para alterar parámetros mientras se mueve el cursor con la tecla Return. No obstante, Yamaha afirma que el software actualizado ya está en camino.

El CX5M está obviamente dirigido a quienes desean un ordenador personal y posean, al mismo tiempo, un interés por la música electrónica. Para estas personas, el paquete que comentamos es, indudablemente, tentador. Uno no sólo obtiene un sistema MIDI completo para ordenador y un teclado, sino que el CX5M ofrece además un potencial mucho mayor que un sistema de precio similar basado, por ejemplo, en el Commodore 64. Parece, no obstante, que el CX5M está destinado a convertirse en un ordenador "culto". El desembolso económico que supone su adquisición es difícil justificarlo ante alguien que no se interese por la música electrónica, puesto que ahora se puede adquirir un ordenador MSX similar a menos de la mitad de precio.

El éxito que obtenga la firma Yamaha con el establecimiento del ordenador como la máquina que deben adquirir los amantes de la música, depende del futuro desarrollo de su software y sus interfaces.

YAMAHA CX5M

DIMENSIONES
413x216x64 mm

CPU
Z80A operando a 3,58 MHz

MEMORIA
48 K de RAM, de los cuales hay 28 K disponibles para programas en BASIC

PANTALLA
Pantalla para textos de 40x24, pantalla para gráficos de 256x192 pixels, con 16 colores y hasta 32 sprites

INTERFACES
Impresora Centronics, TV, monitor compuesto, salida de audio, salida estéreo, 2 puertas para palanca de mando, puerta para cassette, puerta para cartuchos de ROM, bus de ampliación, interface para teclado, puertas para entrada y salida de MIDI

LENGUAJES DISPONIBLES
BASIC, PASCAL, Assembly

TECLADO
67 teclas tipo máquina de escribir con grupo de cursor, más 5 teclas de función programables. El teclado de piano es polifónico de ocho notas con una gama de 3,5 octavas

DOCUMENTACION
El manual es muy irregular. Si bien ofrece algunos detalles sobre cómo utilizar el software para música, no ofrece ninguna explicación de guía. El BASIC MXS apenas si se menciona

VENTAJAS
La máquina ofrece una valor excelente para el músico electrónico. Considerado como un instrumento musical, el CX5M no tiene competencia dentro de su gama de precio

DESVENTAJAS
Será necesario mejorar mucho el software para que la máquina llegue a alcanzar un éxito a largo plazo. El manual es muy limitado y la máquina es demasiado cara para quienes no estén interesados en la música electrónica

Lectura de la Biblia

"'The Word' processor" (El procesador de "La Palabra") es un programa que resalta esclarecedoramente el valor de las grandes bases de datos

"The Word" Processor (El procesador de La Palabra)
para máquinas MS-DOS
Distribuido por
Access Software Ltd, 36 Aybrook Street, London W1M 3JL, Gran Bretaña
Autores
Bible Research Systems, Austin, Texas (Estados Unidos)
Formato
Discos

Ciertas ocupaciones y estudios escolásticos exigen una búsqueda constante a través de grandes cuerpos de texto. Los estudiantes de la Biblia, por ejemplo, requieren de concordancias (índices especializados) para ayudar al estudio de los 39 libros del Antiguo Testamento y los 27 libros del Nuevo Testamento: aproximadamente 750 000 palabras en total.

Los ordenadores por lo general son especialmente indicados cuando es necesario efectuar búsquedas en cantidades tan ingentes de texto, pero a menudo las limitaciones de memoria y capacidades de los discos restringen esta aplicación en los micros personales. Es evidente, entonces, que un programa escrito especialmente podría simplificar muchísimo la tarea de realizar búsquedas de los textos bíblicos apropiados. El programa que nos ocupa, al que se le ha dado el desenfadado nombre de *"The Word" processor* (El procesador de La Palabra), no sólo permite buscar referencias específicas, sino que también construye índices que reducen el tiempo que puede ocupar una búsqueda.

La magnitud de la tarea asumida por los programadores de *"The Word" processor* se hace evidente cuando se abre el paquete. El manual es un delgado folleto de 36 páginas, pero el programa completo, incluyendo archivos de textos bíblicos, se compone de siete discos de doble cara, todos los cuales han de ser copiados. Si se emplea una unidad de una sola cara y 40 pistas, entonces habrá que manipular un total de 14 discos separados, lo que demuestra a todas luces que la manipulación de tantos discos de 5 ¼ pulgadas es demasiado complicada para resultar verdaderamente eficaz.

"The Word" processor requiere una instalación cuidadosa. Es razonable suponer que la mayoría de los estudiantes no son necesariamente apasionados de la informática, a pesar de lo cual el programa en BASIC es bastante engañoso en la forma en que se ejecuta: cuando lo utilizamos, dejó de responder al MS-DOS y respondió sólo al PC-DOS. Las pistas del intérprete de BASIC y del sistema de la máquina del disco DOS se deben copiar en el disco del programa. El estudiante debe entonces elegir entre ejecutar (RUN) el programa principal (TWP), si está utilizando un IBM-PC, otra versión para máquinas compatibles con IBM (ATWP), o una versión especial para ordenadores con sólo 64 K de memoria (TWP64). El usuario ya experimentado supondrá con razón que, una vez instalados en el disco las pistas del sistema, el BASIC y los programas pertinentes, se producirá la autocarga; pero no hay ningún programa AUTOEXEC. BAT con este objeto.

Una vez cargado y en ejecución, la página de títulos nos hace comprender que éste no es un paquete de software común, dado que, además de la nota habitual de protección del software ("Este software está protegido por las leyes de *copyright* de Estados Unidos. Su reproducción o la utilización de copias no autorizadas está penada con prisión o multas de hasta 10 000 dólares"), hay asimismo una cita bíblica muy apropiada:

Exodo 20:15 "No robarás."

Este precepto va seguido por el menú de apertura:

<F1> — VISUALIZAR INSTRUCCIONES DE AYUDA
<F2> — ESTABLECER OPCIONES DE CONTROL
<F3> — LA MODALIDAD DE IMPRESION ESTA APAGADA. ENCENDERLA
<F4> — GAMA:
<F5> — FIN DE ESTA SESION
<F6> — CREAR UN INDICE
<F7> — VISUALIZAR O MODIFICAR INDICE
<F8> — MEZCLAR INDICES
<F9> — ELIMINAR UN INDICE
<F10> — VISUALIZAR TEXTO DE LAS ESCRITURAS

<F2> produce un submenú de nueve opciones de control, incluyendo anchura de pantalla (40 u 80 caracteres), tabulaciones para los márgenes izquierdo y derecho de la impresora, espaciado de líneas, número de líneas por página, dimensiones máximas del índice, visualización de un texto cada vez o una pantalla de textos entera con indicación de los versículos del contexto, y los nombres de todos los índices existentes en el disco actual.

La opción <F4> es una forma de administrar tanto la inmensidad de la Biblia como la proliferación de discos en un sistema basado en discos flexibles. Utilizando una designación de tres letras para cada libro de la Biblia (de *Gen* a *Rev*), y la referencia de capítulo y versículo, uno puede definir el comienzo y el final del parámetro de búsqueda. Si el usuario desea leer un libro específico se pueden omitir las referencias de capítulo y versículo y el programa supondrá que la gama requerida es el comienzo del primer capítulo y el final del último.

Después de haber insertado el disco adecuado, se puede hojear un capítulo (visualizando el primer versículo de la gama mediante <F10>) e ir hacia atrás o hacia adelante mediante las teclas del cursor, que aparecen a través de la parte inferior de la pantalla. La opción de control <F8> cambia la visualización de un versículo cada vez por una pantalla entera de textos, permitiendo leer en su contexto un versículo específico.

El criterio de búsqueda se puede establecer mediante <F2>, la opción SCAN. Tal como sucede con las búsquedas en la mayoría de las bases de datos, se pueden buscar todas las ocurrencias, incluso dentro de otras palabras (de manera que *man* también cogería *demand*), o se puede limitar la búsque-

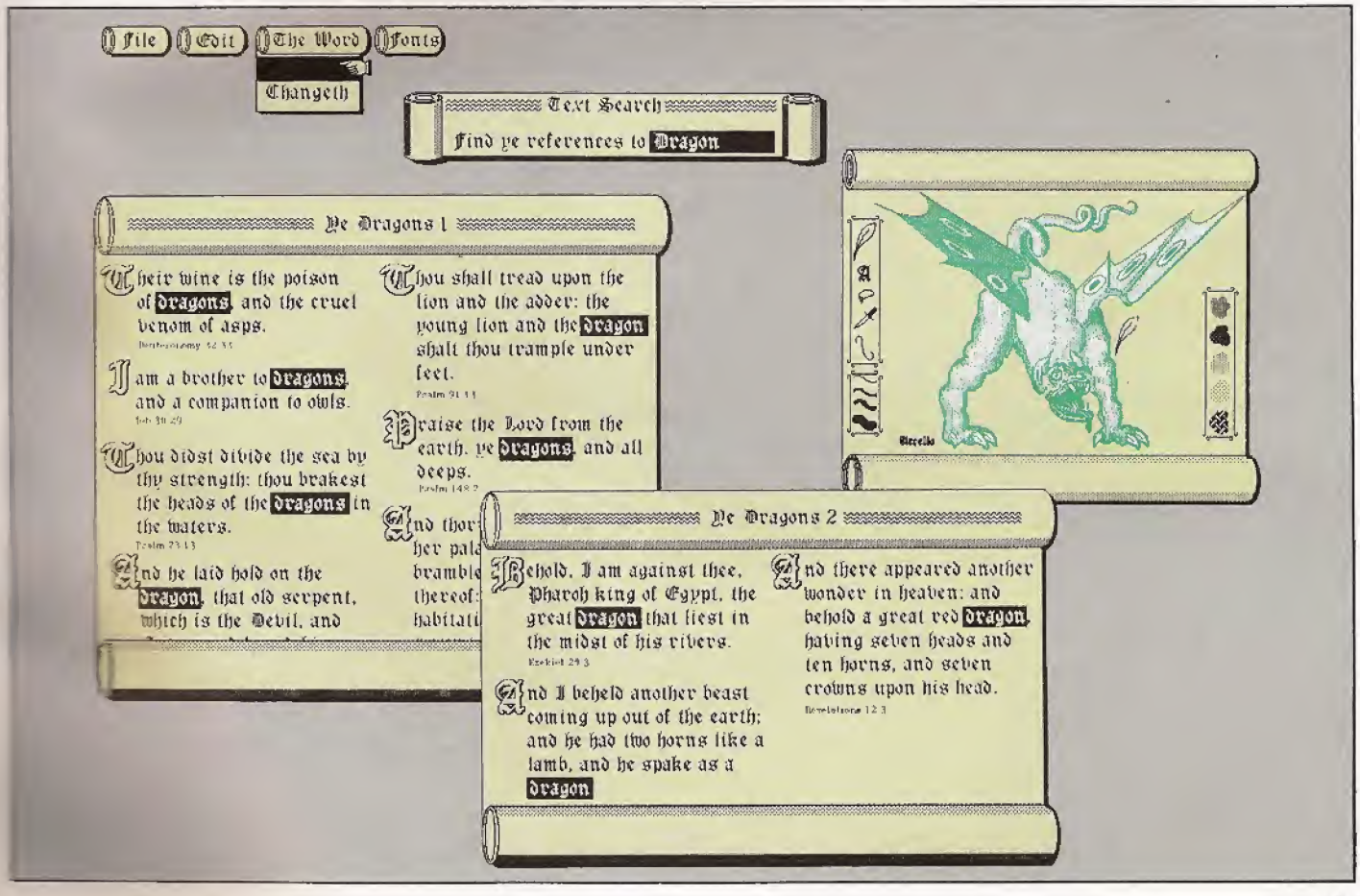
da sólo a un vocablo específico. También se pueden utilizar máscaras, que permiten buscar todas las palabras que empiecen o terminen con una serie de búsqueda especificada. Una vez establecidas, las referencias de todos los criterios se visualizarán de una en una, ya sea en el contexto o individualmente, según lo que se haya solicitado.

Con la opción <F6> se pueden crear índices, ya sea automáticamente o de forma manual. Los índices también se pueden editar (p. ej., suprimiendo referencias no deseadas), y se pueden mezclar dos o más entre sí, siempre que no exceda la cantidad límite de 1 020 elementos en el índice. Este límite se puede aumentar a 53 040 con la opción <F2>, ESTABLECER OPCIONES DE CONTROL, del menú principal. Si un índice posee más de 1 020 referencias, se lo almacena (utilizando el submenú OPCIONES DE CONTROL) por secciones, y entre las entradas 1 020 y 1 021 habrá una pausa cuando la sección siguiente del índice se lea (READ) del disco a la memoria. También se puede crear un índice de los temas que contiene cada versículo. Esto obviamente se ejecuta bajo el control del usuario, quien define el título del tema y el versículo en el cual se pueda hallar.

Además de la dificultad de manipular cantidades tan grandes de texto, el problema principal de *"The Word" processor* reside en que, dado que se basa en la versión del rey Jacobo, que data de 1611, muchas de las palabras utilizadas en esta magnífica traducción han sufrido un cambio radical de significado en los tres siglos y medio posteriores. De acuerdo a los traductores de la Revised Standard Version de 1952, hay más de 300 de tales palabras y frases; y eso sin incluir los errores de traducción que quedaron patentes merced a la aparición de textos más fiables desde el s. XVII.

Afortunadamente, para evitar este inconveniente se puede utilizar un programa, *The Greek transliterator* (El transliterador griego), que será de ayuda al menos en lo que se refiere al Nuevo Testamento. Este programa permite buscar a través del texto griego las apariciones de una palabra dada de la versión del rey Jacobo, solicitando la visualización de la palabra griega original (no en ortografía griega, sino en caracteres romanos) con una definición de la palabra griega y, si fuera necesario, la frecuencia con la que se utilizó tal palabra y los distintos significados que asumió en diversos contextos. Los índices creados con el TWP también se pueden utilizar con *The Greek transliterator*, y también se pueden crear índices de palabras griegas. El precio de *The Greek transliterator* es el mismo que el del TWP.

Hasta la fecha no se ha producido ningún programa transliterador para el Antiguo Testamento, si bien en Jerusalén se ha programado un gran ordenador para almacenar el Talmud y otras obras religiosas. Posiblemente ésta sea la forma más práctica de emplear los estudios informatizados de la Biblia: como una base de datos de acceso público y no como un sistema individual de recuperación de textos. No obstante, las técnicas empleadas en *"The Word" processor* son interesantísimas y, presumiblemente, se podrían aplicar a otras obras a gran escala: la producción escénica y poética de Shakespeare, por ejemplo, o incluso las obras completas de Marx y Engels.



El rey Jacobo y el dragón
Al solicitar *dragon* a través del teclado, la base de datos *"The Word" processor* localizó, entre otras, estas referencias. Actuando como lo haría una concordancia escrita, *"The Word" processor* simplifica la búsqueda de ideas bíblicas relacionadas, símbolos y personalidades, en la versión de la Biblia del rey Jacobo

Ian McKinnell

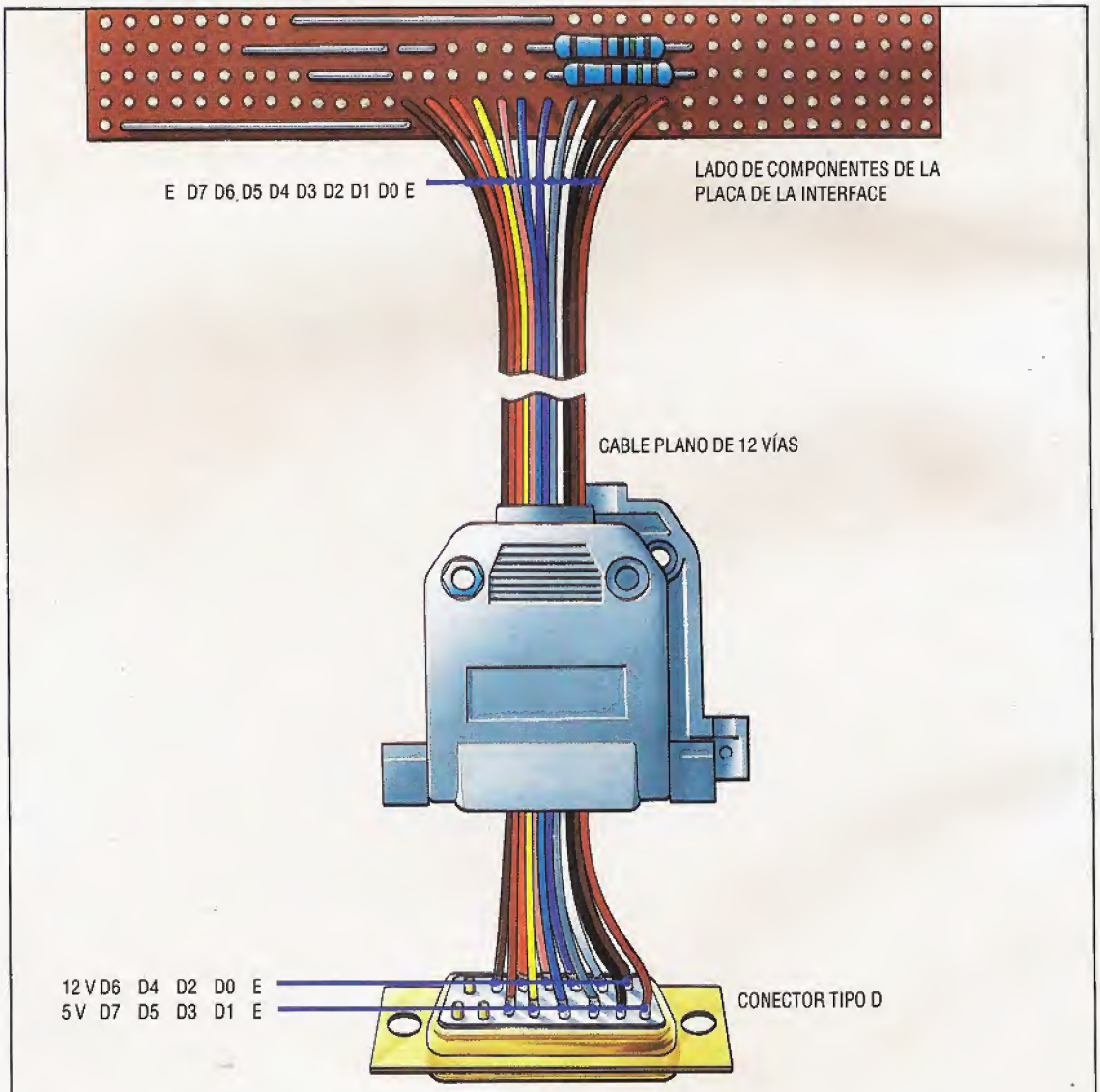
En un cable

Con el fin de controlar el robot, vamos a construir el cable que lo conecta a la interface diseñada en el capítulo anterior

La última tarea en la construcción de la interface es soldar un cable plano de 12 vías desde la placa de la interface hasta un conector tipo D que se enchufará en el robot. Las ocho líneas de datos desde la puerta de E/S están disponibles desde el bus de datos a través del sistema de circuitos de la interface. A la alimentación de potencia de 12 V y 5 V también se accede directamente desde la puerta para ampliación del Spectrum. Ello significa que no tenemos que proporcionar otra fuente de alimentación eléctrica para activar el robot. El suministro de potencia del ZX es capaz de abastecer al ordenador y al robot al mismo tiempo.

Confeccione el cable plano pelando cuatro cables del cable plano especificado en la lista de componentes que ofrecimos para el proyecto. Pele y estañe ambos extremos de cada cabo hacia la placa y el conector tipo D tal como se indica en el diagrama. Compruebe a conciencia su trabajo, asegurándose particularmente de que las alimentaciones de 12 V y 5 V lleguen a las patillas correctas del conector tipo D.

Tendiendo un cable plano de 12 vías
Realice el cableado del conector tipo D con la placa de la interface tal como muestra la ilustración. El cableado es directo: los cables se alternan entre las filas de patillas superior e inferior del conector tipo D



Conexiones de los cables de la interface

La interface está ahora completa y podemos enchufarla en el Spectrum para probarla. Con la potencia apagada, enchufe cuidadosamente la placa de la interface en la puerta para ampliación de la parte posterior del Spectrum. La placa se debe insertar de modo que el lado de los componentes quede arriba; si ha insertado el enchufe anulador en la posición 5 del conector de la puerta para ampliación, será imposible colocar la placa de cualquier otra forma. El encaje de la placa puede ser apretado mediante un ligero movimiento de vaivén, a izquierda y derecha, la ranura del PCB de la puerta de ampliación encajará ceñidamente en el conector de la placa de la interface. Enchufe el cable plano en el robot y encienda la potencia. Si todo está bien, en la pantalla aparecerá el mensaje del *copyright* de Sinclair. Ahora estamos en condiciones de probar un programa de verificación.

Control de motores

El robot se asocia en la puerta de E/S 31, utilizándose sus ocho bits para controlar los motores y recibir entradas provenientes de los sensores del robot. Los cuatro bits inferiores controlan el movimiento. Los bits 1 y 2 controlan la dirección de rotación de los dos motores paso a paso utilizados con el robot. Las cuatro direcciones se pueden seleccionar estableciendo estos dos bits en diversas combinaciones. El bit 3 es el bit de impulsos. Cambiándolo de 0 a 1 ambos motores efectúan un giro de un paso en la dirección que se haya especificado en el correspondiente bit de dirección. El bit 0 es el bit de puesta a cero y normalmente está a 1.

Podemos comprobar la salida de la interface digitando y ejecutando el programa "Controlador del robot", que permite controlar el robot desde el teclado. Las letras T, B, F y H corresponden a adelante, atrás, izquierda y derecha, respectivamente.

La rutina de inicialización establece cuatro variables, que corresponden a las cuatro direcciones posibles del robot. Sus valores se determinan mediante las cuatro combinaciones de los bits 1 y 2 de la puerta de E/S 31. Observando estas variables y sus valores binarios de cuatro bits (véase tabla), podemos ver por qué.

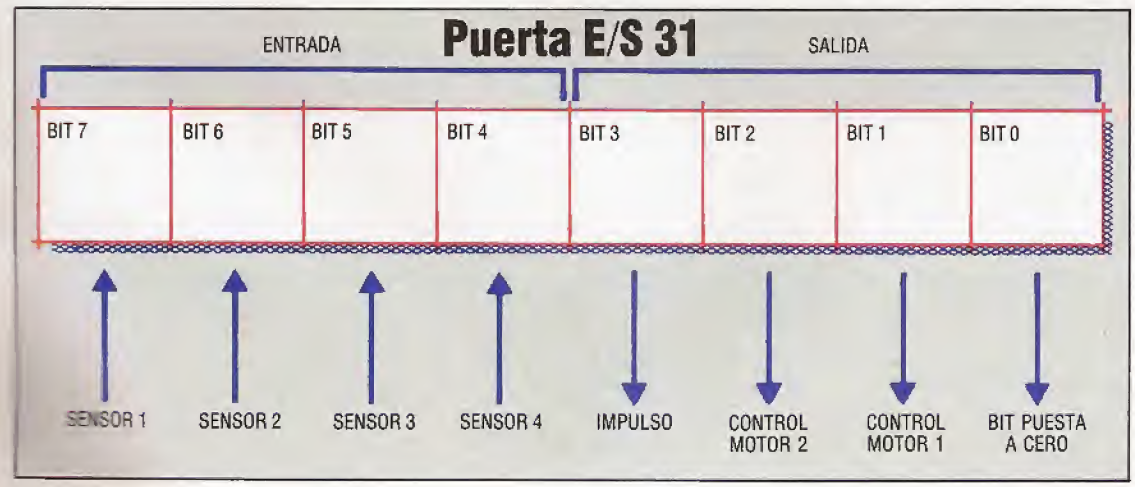
La dirección que toma el robot se retiene en la variable dr. Para conseguir que el robot se mueva en la dirección especificada debemos impulsar los motores estableciendo el bit 3 *high* y luego *low*. Esto lo hace la subrutina de la línea 2000, utilizando OUT, de forma similar a POKE, para colocar un valor en la puerta de E/S 31. El valor en sí mismo se calcula como la dirección actual, dr, más 8 para establecer el bit 3 en 1, es decir, dr+9. Para apagar el bit 3 simplemente no sumamos en el 8, lo que nos da dr+1. Al hacer que el bit 3 se ponga *high* y después otra vez *low*, los motores efectúan un giro de un paso de 7,5°. Como esto corresponde a un movimiento de menos de 1 mm en la rueda, el proceso de impulso se coloca dentro de un bucle para repetirlo un cierto número de veces, determinado por la variable m.

Dirección	Variable	Valor decimal	Valor binario
Adelante	fw	4	0100
Atrás	bw	2	0010
Izquierda	lf	6	0110
Derecha	rt	0	0000

La dirección se altera efectuando una pulsación en el teclado, modificando el valor de dr. Cuando los motores se impulsan la vez siguiente, los bits de dirección del motor cambian haciendo que el motor se mueva en otra dirección.

Detección de la entrada

Los cuatro bits superiores se utilizan para detectar entradas de los sensores del robot, correspondiendo cada bit a un conector del sistema de parches de la tapa del robot. Normalmente estos bits se mantienen *high* (es decir, poseen un valor de 1), a menos que sean dirigidos a tierra mediante el cierre de un interruptor, en cuyo caso el valor del bit pasa a 0. Para detectar las entradas independientes de cada bit necesitamos un método de aislar cada uno de los cuatro bits superiores y comprobar sus valores. En la mayoría de las versiones de BASIC ello es posible utilizando la instrucción lógica AND para enmascarar cualquier bit en el que no estamos interesados. El BASIC Spectrum posee una instrucción AND, pero no es suficientemente sofisticada como para enmascarar los bits individuales de un número. Sin embargo, el juego de instrucciones Z80 contiene una operación AND que realizará la tarea que necesitamos. Por consiguiente, hemos de escribir un trozo sencillo de código de lenguaje máquina que lleve a cabo el AND lógico en dos números y devuelva el resultado. Para emplear la rutina debe-



Los bits de la acción
La interface para el Spectrum establece los cuatro bits inferiores de la puerta de E/S 31 en salida y los cuatro bits superiores en entrada. Los bits de salida controlan las funciones de los motores paso a paso, utilizándose los cuatro bits superiores para aceptar las entradas sensoriales

mos colocar (POKE) los números que deseamos relacionar con AND en dos posiciones de la memoria. El resultado se devolverá mediante la instrucción USR. Este método indirecto nos permite leer cada línea de entrada de forma independiente.

El programa "Parachoques Spectrum" comprueba la acción de la entrada de la interface enviando el robot hacia adelante hasta que encuentre un objeto, retrocediendo entonces a su posición de partida inicial.

Para poner al día a los usuarios del Spectrum, incluimos también los correspondientes *Complementos al* BASIC para algunos de los otros programas que hemos ido creando en la serie de aplicaciones de robot. Los lectores habrán de realizar las alteraciones necesarias sobre las versiones para el Commodore 64. Quizá los usuarios del Spectrum deseen, asimismo, sustituir los nombres de variables en minúsculas por los nombres en mayúsculas utilizados en las versiones Commodore.

Controlador del robot

```
 10 REM **** controlador del robot Spectrum ****
 20 GO SUB 1000:REM inicializar
 30 LET a$=INKEY$:IF a$<>"x" THEN GO SUB
    3000:REM leer tecla
 40 LET m=10:GO SUB 2000:REM impulso
 50 IF a$<>"x" THEN GO TO 30
 60 OUT 31,0
 70 STOP
 80 :
1000 REM **** inicializar ****
1010 LET fw=4:LET bw=2:LET lf=6:LET rt=0
1020 LET dr=fw
1030 RETURN
1040 :
2000 REM **** impulso ****
2010 FOR c=1 TO m
2020 OUT 31,dr+9
2030 OUT 31,dr+1
2040 NEXT c
2050 RETURN
2060 :
3000 REM **** leer teclas ****
3010 IF a$="t" THEN LET dr=fw:RETURN
3020 IF a$="b" THEN LET dr=bw:RETURN
3030 IF a$="f" THEN LET dr=lf:RETURN
3040 IF a$="h" THEN LET dr=rt:RETURN
3050 RETURN
```

Parachoques Spectrum

```
 10 REM **** parachoques spectrum ****
 15 CLEAR 32499:LET st=32500
 20 GO SUB 500:REM inicializar
 25 GO SUB 3000: REM cargar lenguaje maquina
 30 LET dr=fw
 40 REM **** impulso adelante ****
 50 LET cc=0
 60 GO SUB 1000:LET cc=cc+1:REM impulso
 70 REM ** comprobar parachoques **
 80 LET nm=192:GO SUB 2000: REM efectuar AND
 90 IF USR st=192 THEN GO TO 60
100 REM **** vuelta al comienzo ****
110 LET dr=bw
120 FOR i=1 TO cc
130 GO SUB 1000:REM impulso
140 NEXT i
150 OUT 31,0:STOP
500 REM **** inicializar ****
510 LET fw=4:LET bw=2:LET lf=6:LET rt=0
520 RETURN
1000 REM **** impulso ****
1010 OUT 31,dr+9
1020 OUT 31,dr+1
1030 RETURN
2000 REM **** efectuar AND ****
2010 POKE st+1,IN 31
2020 POKE st+3,nm:RETURN
3000 REM **** cargador lenguaje maquina ****
3010 FOR i=st TO st+8
3020 READ a:POKE i,a
3030 NEXT i
3040 DATA 62,0,14,0,161,6,0,79,201
3050 RETURN
```

Complementos al BASIC

En el programa Calibrado Lineal suprima las líneas 20, 30 y 70 e introduzca los siguientes cambios:

```
 80 LET dr=fw
100 LET a$=INKEY$:IF a$=""THEN GO TO 100
280 OUT 31,dr+9
290 OUT 31,dr+1
```

En el programa Calibrado Angular suprima las líneas 20, 30 y 50 e introduzca las siguientes modificaciones:

```
 60 LET dr=rt
 80 LET dr=fw
100 OUT 31,dr+9
110 OUT 31,dr+1
```

En el programa Medición Robot suprima las líneas 1010, 1020, 1030, 7010 y 7510 e introduzca las siguientes modificaciones:

```
2050 GO SUB 8100:IF USR st<>lb THEN GO TO
     2040
2090 GO SUB 8100:IF USR st<>lb THEN GO TO
     2080
2150 GO SUB 8100:IF USR st<>rb THEN GO TO
     2140
2190 GO SUB 8100:IF USR st<>rb THEN GO TO
     2180
2200 OUT 31,0
3010 CLS
3520 GO SUB 8100:IF USR st=nb THEN GO TO
     3510
4010 GO SUB 8100:IF USR st=rb THEN LET
     ss=rt: GO SUB 5000:RETURN
4020 GO SUB 8100:IF USR st=lb THEN LET
     ss=lf: GO SUB 5000:RETURN
5020 GO SUB 8100:IF USR st=nb THEN GO TO
     5010
6080 GO SUB 8100:IF USR st=nb THEN GO TO
     6070
8100 REM **** efectuar AND ****
8110 POKE st+1,192:POKE st+3,IN 31:RETURN
8200 REM **** cargador lenguaje maquina ****
8210 FOR i=st TO st+8
8220 READ a:POKE i,a
8230 NEXT i
8240 DATA 62,0,14,0,161,6,0,79,201
8250 RETURN
```

Viejos y nuevos

Se inicia aquí un estudio sobre las relaciones entre el BASIC y el sistema operativo en el BBC Micro

La ROM para el BASIC en el BBC Micro ocupa la zona de memoria situada entre las direcciones &8000 y &BFFF. Esta zona se conoce también como espacio paginado de la ROM. Quiere decir que, mediante un sabio empleo del direccionamiento paginado, pueden corresidir en la máquina varios chips de ROM y emplear el mismo bloque de memoria. Sólo una de estas ROM está en activo, o sea "paginada", en cada ocasión. Son ejemplos de ROM que ocupan esta área de la memoria el chip DFS, el Econet, utilidades del tipo Wordwise, View o Disk Doctor, y otros lenguajes como FORTH y BCPL. La ventaja que reporta este empleo de ROM paginadas es obvia: ahorra memoria.

El que esté en activo una u otra ROM paginada depende del valor de *registro paginador (paging register)*, que forma parte del hardware controlado por el OS. Es posible acceder a un máximo de 16 ROM paginadas al mismo tiempo. El OS examina estas ROM cuando sucede algo inesperado. Por ejemplo, si una instrucción * no es reconocida por el OS, éste interroga a las ROM presentes en la máquina antes de dar el mensaje BAD COMMAND. Por ejemplo, *INFO no será recibida en la ROM del OS y será pasada a alguna de las ROM existentes en la máquina. Si está colocada, la ROM del DFS la aceptará y actuará en consonancia.

Versiones del BASIC del BBC

Dos versiones del BASIC han sido diseñadas para los micros BBC empleados en Gran Bretaña. Son el BASIC I y el BASIC II. Basta con pulsar CTRL-Break, digitar REPORT y pulsar finalmente Return para saber qué versión es la que alberga un micro determinado. Si se imprime un mensaje de *copyright* fechado en 1982, la versión es BASIC II. Si la fecha es 1981, se trata de la versión más antigua, el BASIC I. La nueva versión eliminó algunos de los gazapos que se deslizaron en la vieja y añadió otras características.

Estas nuevas características incluyen la instrucción de archivo, OPENUP, y cambian el papel desempeñado por OPENIN. En el BASIC II, OPENIN abrirá un fichero sólo para entrada; en el BASIC I la apertura de un fichero puede servir, con esta instrucción, tanto para entrada como para salida, lo que constituye una característica valiosa para los ficheros en disco de acceso directo. OPENUP en el BASIC II abre un fichero tanto para entrada como para salida, y puede considerarse el equivalente del antiguo OPENIN del BASIC I. Este cambio produce las complicaciones propias de las transferencias de programas entre máquinas. Si usted escribió un programa que incluía instrucciones OPENIN en BASIC II, a la hora de cargarlo en una máquina con BASIC I se verá en un buen lío. El símbolo *(token)* que el BASIC II emplea para representar OPENIN no es reconocido por el BASIC I. Por ejemplo, esta sentencia:

```
Y%=OPENIN("RAUL")
```

en BASIC II, se lee

```
Y%=("RAUL")
```

en BASIC I, dando el consiguiente mensaje de error.

Las restantes facilidades del BASIC II tienen un interés especial para los programadores en lenguaje máquina, y en particular hay que mencionar una mejora importante. En el BASIC I si se requería la incorporación de datos a los programas en código máquina, se tenía que abandonar el ensamblador del BBC y usar los operadores ?, ! y $ para guardar los datos en la memoria.

Por ejemplo:

```
1000 LDA dato
1010 .dato
1020 ]
1030 ?P%=00
1040 P%=P%+1
1050 OPT pass
```

Lo que resulta algo complicado, y muchos ensambladores nos ofrecen un camino que lo evita por medio de *directivas de ensamblador*. Se trata de instrucciones que indican al ensamblador la tarea concreta que ha de realizar: en nuestro programa OPT es una directiva de ensamblador. Pero el BASIC II proporciona cuatro directivas de ensamblador más: EQUB guarda un byte particular en la memoria; EQUD guarda dos; EQUW cuatro y EQUS permite almacenar una serie en la memoria. El programa anterior puede escribirse así, en BASIC II:

```
1000 LDA dato
1010 .dato EQUB 00
1020 ...resto del programa
```

Esto hará que el registro A se cargue con el byte contenido en la dirección dato, que en este ejemplo es cero.

Otro ejemplo sería:

```
1000 .mensaje EQUS "Hola chicos"
```

En este caso, los bytes que representan el mensaje Hola chicos se colocarán en la memoria en la dirección etiquetada .mensaje. Cuando se emplean todas estas directivas de ensamblador, P% se actualiza automáticamente teniendo en cuenta los datos.

Otro añadido útil del BASIC II es la instrucción OSCLI. Ya comentamos cómo se pasan las instrucciones * al OS del BBC mediante la llamada OSCLI en la dirección &FFF7. Con el BASIC I había que establecer las variables X% e Y% (o los registros X e Y) con la dirección en memoria de la serie que representaba la instrucción que había que pasar. La

nueva instrucción OSCLI en el BASIC II hace mucho más fácil la tarea. Comparémoslas:

BASIC I	BASIC II
$&A00="*TAPE" X%=&A00 MOD 256 Y%=&A00 DIV 256 CALL &FFF7	OSCLI ("*TAPE")

Además de estas dos versiones del BASIC, existe un BASIC II modificado para el mercado estadounidense; las alteraciones de esta versión sólo hacen referencia a las características de la visualización. También el segundo procesador 6502 contiene su propia versión del BASIC II llamada HI BASIC. No parece sino una mezcla de las versiones estadounidense y del BASIC II propiamente dicho. Desde el punto de vista del programador es una versión idéntica al BASIC II. Finalmente, añadiremos que ambas versiones del BASIC emplean del mismo modo la memoria del BBC Micro.

Espacio de trabajo para el lenguaje

El área de memoria comprendida entre &400 y &7FF se conoce como *espacio de trabajo para el lenguaje (language workspace)*; se reserva al lenguaje que actualmente se está usando. Además, la página cero entre las direcciones &00 y &6F queda reservada para ese mismo lenguaje en uso, aunque parte de dicho espacio, el comprendido entre &50 y &6F no parece que sea muy utilizado. La página 4 de la memoria la emplea el BASIC para el almacenamiento de las variables; de &400 a &46B es el área de memoria donde los valores asignados a las variables enteras residentes quedan almacenados (variables tales como A% o X%). La variable @% se almacena en los cuatro bytes que van del &400 al &403 (con el byte menos significativo en &400), mientras que A% se almacena desde &404 a &407, y así sucesivamente hasta la última variable, que es Z%.

El área de memoria entre &480 y &4F9 sirve de índice para el intérprete BASIC; apunta a los lugares de la memoria donde se almacenan otras variables enteras o de coma flotante, matrices o series. Por ejemplo, los dos bytes de &482 y &483 apuntan a un área de la memoria (entre TOP y HIMEM) donde pueden encontrarse detalles de todas las variables actualmente empleadas cuyos nombres comiencen por la letra A. La dirección guardada en las posiciones &4F6 y &4F7 se refiere al almacenamiento de los procedimientos del BASIC, y la dirección guardada en las posiciones &4F8 y &4F9, al almacenamiento de las funciones.

El área de memoria entre &500 y &5FF sirve de pila para el intérprete del BASIC. Lo que no ha de confundirse con la pila del 6502 almacenada en la página 1 de la memoria. La página 5 la emplea el BASIC para almacenar los detalles de RETURN para toda llamada GOSUB, y la información necesaria para la ejecución de los ciclos FOR...NEXT y REPEAT...UNTIL.

La página 6 de la memoria sirve al BASIC o bien para el tratamiento de variables de serie o bien para el caso que se emplee una instrucción CALL que permita ejecutar un programa en código máquina, de lo que se hablará más adelante. El BBC BASIC utiliza la página 7 como buffer de entrada para cuando se digitan instrucciones BASIC o líneas de programa desde el teclado. Todo lo que se digita se sitúa en esta área de la memoria en espera de ser procesado. Si es una instrucción directa, se interpreta y se ejecuta; si es una línea de programa se coloca en la posición correspondiente dentro del programa. Concluimos así este repaso detallado del uso que el BASIC hace del espacio de trabajo para el lenguaje en un BBC Micro. Algo que resulta de utilidad en esta área de memoria es que si se deja de emplear el BASIC, queda disponible para cualquier otro uso.

Tratamiento de errores en BASIC

Quién más quién menos todos cometemos errores en nuestros programas en BASIC, y hemos experimentado cómo nos responde el ordenador con sus mensajes de error. Lo que es bastante interesante en este tema del tratamiento de los errores en el BBC Micro es que podemos aprovechar el "manejador de errores" para elaborar nuestras propias sentencias de error, completándolas con mensajes y números. Esto es muy útil cuando conjuntamos programas en código máquina con otros en BASIC. La rutina encargada de tratar los errores generados por un programa BASIC tiene su dirección en el contenido del vector llamado BRKV, el cual se sitúa en la dirección &202 y &203. Veamos cómo se da entrada a la rutina a la cual apunta este vector.

Cuando el BASIC tiene que generar un error porque ha ocurrido algo ilegal (p. ej., dividir por cero), éste ejecuta una instrucción del 6502 en lenguaje máquina llamada BRK. Lo que produce un salto a la rutina especificada en la dirección contenida en BRKV (*break vector*: vector de ruptura). En condiciones normales esta rutina espera encontrar la instrucción BRK seguida de una serie de bytes, y establecida de la siguiente manera:

BRK
Número del error (un byte)
Mensaje de error (serie de bytes)
Fin del mensaje (un byte cero)

Por ejemplo, para el error número 4, que genera el mensaje *Mistake* (equivocado), se ejecutaría la siguiente rutina para hacer entrar al manejador de errores.

Byte	Comentario
BRK	Instrucción
4	Número de error
M	
i	
s	Código ASCII
t	del mensaje
a	de error
k	
e	
00	Fin del mensaje

El manejador de errores del BASIC establece entonces la variable de sistema ERL para que guarde el

número de línea en que ocurrió el error. Seguidamente se coloca el número del error en ERR (en nuestro caso 4) y el manejador se encarga de que a continuación se evalúe REPORT y que se imprima el mensaje *Mistake*.

La instrucción BRK se puede incluir en los programas en código máquina, por lo que podemos elaborar nuestros propios mensajes de error. Cualquier mensaje de error que generemos se aceptará en BASIC mediante la instrucción ON ERROR GOTO, como si el error lo generara el intérprete del BASIC. Por ejemplo, el siguiente fragmento de un programa en lenguaje máquina proporciona el mensaje de error *Number too big* (número demasiado grande):

```
1010 BRK
1020 EQUB 254
1030 EQUS "Number too big"
1040 EQUB 00
```

Una vez ejecutado, si imprimimos ERR nos dará 254 y REPORT dará *Number too big*.

Es posible alterar el contenido de este vector y cambiar el modo en que el BBC Micro responde a los errores. Empleando este vector se pueden añadir nuevas instrucciones al BASIC del BBC, pero esto requiere una técnica de programación algo compleja.

En el próximo capítulo de este curso examinaremos la forma en que se produce la interacción entre el BASIC y el sistema operativo del BBC Micro.

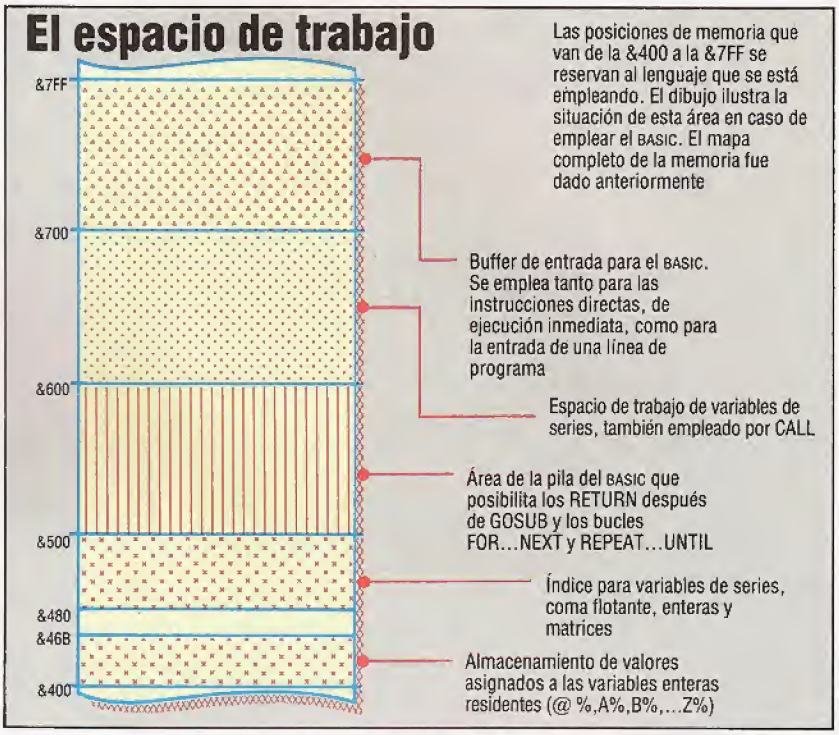
Repaso de rutinas

Vamos a dar una lista de las direcciones en la ROM de algunas de las rutinas en BASIC del BBC Micro que hasta ahora se han mencionado. El examen de dichas rutinas le hará adquirir mayor conciencia del modo de operar del BASIC. No todas las aquí reseñadas terminan con RTS, por tanto no es recomendable su llamada indiscriminadamente desde programas propios, pero el examen de la técnica empleada puede ayudarnos a mejorar la propia habilidad de programación.

Punto de entrada a la ROM	**&8000**
Sentencia BGET	&BF6F
Sentencia BPUT	&BF58
Sentencia CHAIN	&BF2A
Sentencia CLOSE	&BF99
Función EOF	&ACB8
Función INKEY	&ACAD
Función INKEY$	&BO26
Sentencia INPUT	&BA44
Sentencia INPUT#	&B9CF
Instrucción LOAD	&BF24
Función OPENIN	&BF78
Función OPENOUT	&BF7C
Función OPENUP	&BF80
Sentencia PRINT#	&8D2B
Instrucción SAVE	&BEF3
Sentencia TIME	&92C9
Función TIME	&AEB4
Sentencia VDU	&942F

La información es por cortesía de Acorn Computers. © Acorn Computer Ltd (1982)

El espacio de trabajo



Liz Dixon

Variantes del BASIC I y BASIC II en el BBC

Instrucción o error	BASIC I	BASIC II
REPORT	Da el mensaje 1981	Da el mensaje 1982
OPENUP	No implementada	Abre un fichero para lectura y escritura (como OPENIN en el BASIC I)
OPENIN	Abre fichero para lect. y escr.	Abre fichero para lectura solamente
OPT n	OPT 1 hasta 3 son las únicas implementadas	Se conocen OPT 1 hasta 7. Las cuatro nuevas instrucciones equivalen a las otras tres, pero el código se ensambla a la dirección contenida en la variable O%
O%	Sin uso práctico	Función diseñada en una entrada previa (instr. OPTn)
EQUB, EQUD, EQUS, EQUW	No implementadas	Permite al programador incluir datos en programas en código máquina sin abandonar el ensamblador (como vimos)
INSTR (a$,b$)	Grave error: se producía un crac cada vez que a$ era más corta que b$	Corregido
Núms. reales	Almacenados con precisión de 9 cifras	Almacenados con precisión de 10 cifras
OSCLI	No implementada	Posibilita el paso de instrucciones al OS desde variables de series del BASIC

Comercio cósmico

Comienza la misión

Diagrama de corto alcance

Estación espacial

Comportamiento "elitista"
En las fotografías vemos tres escenas de *Elite*, incluyendo la pantalla de apertura, que representa a la nave Cobra Mark III. Completar el juego puede llevar meses y, por consiguiente, posee la facilidad de guardarlo (SAVE) en cualquier punto de su desarrollo. En la parte inferior de la pantalla aparecen los diversos indicadores necesarios durante el viaje y un mapa tridimensional por radar de la zona.
La segunda pantalla muestra las estrellas próximas a su posición actual. Desplazando el punto de mira a una estrella escogida, aparecerá una pantalla que proporciona datos acerca del planeta: su forma de gobierno, tipo de economía, etc. La tercera pantalla muestra la estación espacial. Para realizar el acoplamiento el jugador debe conducir su nave hasta un pequeño puerto rectangular

Acornsoft promete premios en efectivo y certificados para quienes logren las mayores puntuaciones en su juego "Elite"

Elite, de Acornsoft, combina elementos de los juegos recreativos, de aventuras y de estrategia. El objetivo del juego consiste en pasar a formar parte de la Élite. Para conseguirlo, el jugador se debe convertir en un mercader con éxito entre los miles de planetas de ocho galaxias.

Para permanecer con vida el tiempo suficiente para amasar fortuna y unirse a la Élite, el jugador no sólo debe convertirse en un hábil navegante y guerrero, sino que también necesitará buenas aptitudes para los negocios, cierta dosis de astucia y ser capaz de moverse por la difusa línea que separa lo legal de lo ilegal. Se pueden obtener enseguida enormes beneficios haciendo contrabando entre los planetas o convirtiéndose en pirata, pero el jugador ha de tener cuidado. Es posible que la policía se entere de sus actividades y podría verse catalogado como delincuente o fugitivo... y las veloces naves de la policía disparan a matar.

El juego comienza con el jugador acoplado en una estación espacial en la superficie del planeta Lave. Su nave es una Cobra Mark III; está equipada con un láser de impulso frontal y posee una capacidad de carga de 20 toneladas. Antes de partir, es aconsejable familiarizarse con los muchos controles y opciones disponibles, así como practicar las maniobras de acoplamiento ya que, de lo contrario, uno puede estrellarse en su primera misión. Si usted pierde demasiado tiempo intentando acoplarse a una estación espacial u orbitando alrededor de un planeta hostil, seguramente atraerá la atención de los piratas.

Una vez realizado el acoplamiento, llega la hora de hacer negocios. Comparando los precios del planeta con las mercancías que usted posee, debe decidir si le conviene más vender o conservarlas hasta hallar condiciones más favorables en algún otro sitio. Uno de los pocos inconvenientes del juego, al igual que ocurre en la mayoría de los otros juegos de comercio existentes para ordenadores, es que no hay ninguna opción para el regateo. El jugador debe aceptar el precio ofrecido o bien rechazarlo.

El verdadero secreto del comercio consiste en decidir qué productos llevar a cada planeta. Por ejemplo, se puede comprar alimentos a bajo precio en un planeta agrícola, y venderlos a buen precio en un planeta industrial. Luego se puede llevar maquinaria al país agrícola. El jugador debe también tener en cuenta la estructura política del planeta y el tipo de seres que lo habitan. Una estructura política caótica significa que es probable que haya piratas, y algunas especies de alienígenas tienen más tendencia a estafarlo o eliminarlo que otras.

Los gráficos tridimensionales de *Elite* son excelentes. Los planetas, las estaciones y las naves espaciales están dibujadas de forma esquemática y acostumbrarse a ellas cuesta un poco al principio, pero uno comienza enseguida a apreciar la ventaja de una resolución más alta y la capacidad de girar en las tres dimensiones. Esto se aprovecha al máximo durante los acoplamientos y los combates. Las estaciones espaciales sólo poseen una única entrada y la aproximación se debe realizar en ángulo recto. Cuando uno entra en combate, la capacidad de ver la nave enemiga (de la cual hay muchos tipos diferentes) le añade al juego un gran realismo.

Además de la cassette o el disco se suministra un exhaustivo manual que incluye detalles relativos a todos los aspectos del juego, una historia basada en el juego llamada *The dark wheel* (La rueda oscura), una ficha que contiene un resumen de los controles, una plantilla para las teclas de función y un *poster* que ilustra las diversas clases de naves espaciales con la que es probable que se encuentre.

Elite: Para el BBC Micro y el Electron (palancas de mando opcionales)
Editado por: Acornsoft Ltd, Betjemin House, 104 Hills Road, Cambridge, CB2 1LQ, Gran Bretaña
Autores: Ian Bell, David Braben
Formato: Cassette o disco

Necesidades vitales

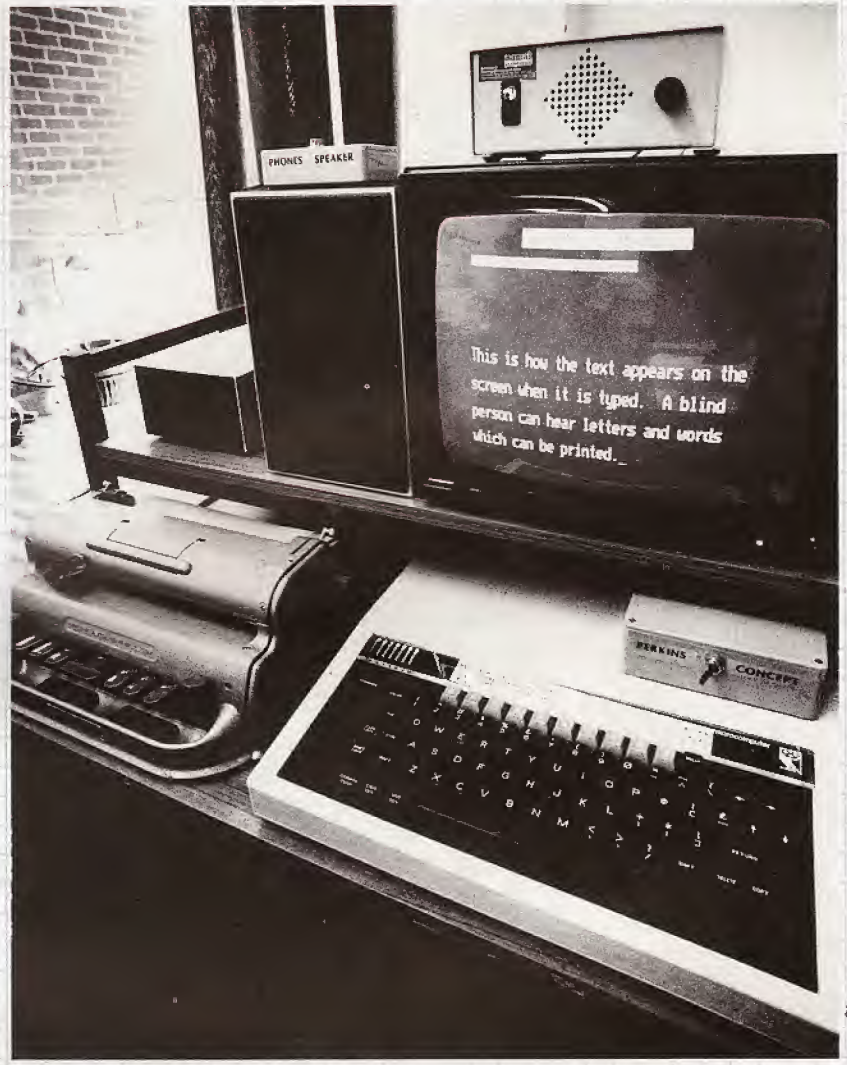
Para los minusválidos el microordenador constituye un medio vital para superar las barreras del mundo de la normalidad

Muchos niños y adultos tienen necesidades especiales desde el punto de vista de la educación. Ello puede deberse a una minusvalía física como la sordera o la ceguera, o a una deficiencia mental. Si bien hay algunos requerimientos que son obvios, hay otros que son difíciles de diagnosticar y comprender y cada incapacidad presenta sus propios problemas, a menudo originados por la insensibilidad de la sociedad más que por la incapacidad en sí.

Lamentablemente, los recortes presupuestarios practicados por algunos gobiernos inciden tan negativamente en las pocas escuelas especiales que existen como en las de educación general. Por consiguiente, éste no es un mercado lucrativo para quienes desarrollan hardware y software. No obstante, la energía y la dedicación de quienes trabajan en este campo pueden servir de ejemplo para el resto del sistema educativo. Un grupo de voluntarios de South Yorkshire, en Gran Bretaña, por ejemplo, reunió 10 000 libras esterlinas en su tiempo libre para desarrollar un teléfono especial de bajo costo para los sordos. Gran parte de este tipo de trabajo lo realizan a nivel de "comunidad" personas individuales y grupos de voluntarios, y a lo largo y ancho de las islas hay maestros de escuela, programadores e ingenieros que están trabajando por iniciativa propia, desarrollando hardware y software para educación especial.

En Gran Bretaña se han abierto SEMERC (*Special Education Microelectronics Resources Centres*: centros de recursos microelectrónicos para la educación especial) en Manchester, Newcastle y Redbridge, para satisfacer las necesidades de la educación especial. Una de sus principales funciones es educar a los maestros de escuelas especiales e informarles de los desarrollos pertinentes. Los recursos de los cuatro SEMERC son pequeños y su labor es muy amplia. El centro de Redbridge atiende a 700 escuelas y debe dividir su tiempo entre educar al personal, dar consejos e información, y evaluar y desarrollar hardware y software.

La situación se complica aún más por los muchos problemas diferentes de la educación especial. En muchas incapacidades, por ejemplo, es la comunicación lo que ofrece los problemas más graves. Una persona inteligente y vivaz puede estar atrapada en un cuerpo incapacitado, imposibilitada de comunicarse con el mundo exterior. La microtecnología ha abierto nuevos canales de comunicación para muchas personas. El Photonic Wand, por ejemplo, es un sensor óptico montado en un casco plástico y controlado por los movimientos de la cabeza, y permite que las personas que no pueden hablar y no tienen control sobre sus extremidades puedan operar un BBC Micro. El aparato se conecta al ordenador a través de la puerta para entrada analógica. El sensor óptico es similar a un lápiz óptico y mueve un cursor a través de la pantalla en respuesta a los movimientos de la cabeza.



Tony Steep

Con este aparato se puede usar un programa para tratamiento de textos denominado *Write*. Visualiza el alfabeto en la pantalla y pueden seleccionarse letras individuales, en mayúscula o minúscula, señalándolas con el cursor. Dispone de una función de edición y de listas de palabras. El texto se puede imprimir, salvar o suprimir. Un programa llamado *Paint* permite dibujar con seis colores, y otro, llamado *Music*, visualiza un teclado. Las notas a tocar se seleccionan con el cursor.

Este dispositivo lo pueden utilizar personas con movimiento inestable de la mano y niños a partir de ocho años. En el SEMERC de Manchester se han llevado a cabo trabajos para conectar la Photonic Wand a un sintetizador de voz. Se seleccionan de la pantalla respuestas diseñadas para conversaciones

Lugar de trabajo
El puesto de trabajo Vincent combina numerosas modificaciones que lo hacen muy útil para personas con diversas incapacidades. El Perkins Brailler permite que los invidentes entren información en el ordenador, y el sintetizador de voz proporciona la realimentación audible. La visualización de textos en la VDU se ha agrandado para beneficiar a los parcialmente invidentes, y el puesto de trabajo también está conectado a un teclado conceptual. Si bien el puesto de trabajo reflejado aquí tiene acoplada una impresora normal, también existe una impresora Braille especial, pero su precio la hace poco asequible

telefónicas, tales como "¿puedes repetir esa frase?", que son pronunciadas por un sintetizador de voz de D.E. Systems. Se espera ampliar el vocabulario y desarrollar una forma sencilla de conectar el sintetizador al teléfono.

Los ordenadores se diseñan para las personas que pueden ver. Las pantallas, las impresoras y los plotters (trazadores de gráficos) son todas salidas visuales y la totalidad de los programas se valen de las visualizaciones. El proyecto Computing and the Blind (La informática y los ciegos) de la Open University ha estado investigando formas de adaptar el hardware de bajo costo existente para que lo utilicen los invidentes. El proyecto se ha puesto en marcha en colaboración con los ciegos en la escuela, en el trabajo y en casa. En las escuelas y centros laborales se han instalado puestos de trabajo que constan de un microordenador BBC, unidad de disco, pantalla, impresora, sintetizador de voz, teclado conceptual y un Perkins Brailler modificado.

El Perkins Brailler se inventó hace 40 años para mecanografiar en braille, sistema de escritura que consiste en un código de puntos repujados sobre papel que permite la lectura al tacto. Se ha conectado en interface con el ordenador para que un invidente pueda leer la salida impresa. Se dispone de software para que el ordenador transcriba el braille a texto normal y pueda almacenarlo, editarlo e imprimirlo.

Se han desarrollado varios procesadores de textos parlantes para ser usados con el teclado estándar. Los caracteres digitados y las teclas especiales se verifican mediante la salida hablada. La tecla Delete también pronuncia el nombre del carácter borrado. El texto se edita utilizando un cursor de voz. A medida que éste se va desplazando a través del texto, se van pronunciando los caracteres, palabras o frases. El cursor se puede detener para añadir o suprimir texto. Se definen márgenes, cabeceras y espacios para producir una copia final pulcra, imposible de diferenciar de una producida por una persona vidente.

Algunos niños incapacitados no poseen la suficiente coordinación física como para utilizar un teclado estándar. Pensando en estas personas se ha desarrollado un teclado plano sensible al tacto, denominado *teclado conceptual*, que permite colocar encima cubiertas intercambiables diseñadas por el maestro. El teclado, por ejemplo, se puede dividir en cuatro secciones y dibujar una imagen en cada cuarto, posibilitando el control de una tortuga para el suelo. Pulsando diferentes secciones se puede dirigir la tortuga hacia adelante, atrás, izquierda y derecha. Lo cual es más sencillo que tener que digitar FD 1 RETURN. Para el teclado conceptual existe algo de software y un lenguaje denominado Starset.

El "ratón" que se utiliza con el Apple Macintosh también evita la "barrera del teclado" y posee un enorme potencial para la educación especial.

Se han desarrollado interruptores especiales, operados mediante diversas partes del cuerpo. Por ejemplo, se ha diseñado un interruptor para personas que carecen de movimiento voluntario. Dos pequeños discos metálicos colocados sobre la piel junto a los ojos detectan el movimiento horizontal del globo ocular y las señales eléctricas se amplifican sobre unos interruptores de control.

En el Departamento de Ciencia y Sociedad de la Universidad de Bradford se le instaló a una tortuga una luz naranja intermitente, que podía ser vista por un incapacitado físico de 18 años de edad. Utilizando un teclado conceptual, el adolescente fue capaz de desplazar la tortuga por el suelo y ver los resultados de sus acciones. Para este adolescente, así como para muchos niños incapacitados, experiencias como ésta suelen representar el primer paso para dejar de ser un observador pasivo de su entorno y comenzar a influir conscientemente en el mismo.

Un toque mágico
La banda fotónica permite que incluso personas gravemente incapacitadas, imposibilitadas para hablar y mover sus extremidades, puedan controlar un BBC Micro. El movimiento de la banda se detecta mediante un sensor óptico y se traduce a una señal analógica

Chris Barker, cortesía de Educational Computing Magazine

En el congreso del British LOGO User's Group celebrado en 1984, la doctora Sylvia Weir ofreció notables ejemplos del progreso que experimentaron niños incapacitados mental y físicamente gracias al empleo del LOGO y las tortugas. Michael, de 17 años y alumno de una escuela especial, jamás había escrito ni una palabra. Sus maestros intuían que era inteligente y que nunca había tenido ocasión de demostrarlo. Tras introducirlo en la informática, al poco tiempo se pasaba diez horas diarias programando. Dos años después estaba en la universidad escribiendo un trabajo titulado *Inteligencia prisionera*. A un niño autista de siete años de edad se le permitió controlar una tortuga desde una caja de botones, un dispositivo similar al teclado conceptual. Se mostró tan entusiasmado con la experiencia, que por primera vez comenzó a hablar.

La doctora Weir también ha empleado el LOGO en una escuela especial con un grupo de adolescentes afectados de parálisis cerebral. Ha escrito: "La notable mejora resultante, en términos del sentido de la valía personal y las consecuencias de la actividad intelectual, han llevado a la escuela a implementar su propio centro de informática, utilizando un sistema de ordenadores recientemente implementado."

Un esfuerzo conjunto del SEMERC de Manchester y un club de informática local produjo el Micromike, un dispositivo para ayudar a los niños con defectos de habla. Un micrófono de radio CB modificado que se conecta a un microordenador BBC, permite al niño controlar diversas actividades en la pantalla utilizando su voz. El software desarrollado para el mismo incluye *City*, programa que permite dibujar la línea del horizonte de una ciudad. La altura y la anchura de los edificios se determinan mediante el volumen y la duración de la voz. Otros programas permiten representar estrellas en el cielo, conducir un barco a través de los rápidos y conseguir que un helicóptero rescate a un piragüista. Todos ellos le ofrecen al niño la oportunidad de controlar su voz de una forma única.

Al igual que la educación normal, la educación especial está a la espera del desarrollo tecnológico. Carece de fondos como para comprar y en el mundo comercial es muy poca la investigación y el desarrollo que se realizan para ella, porque las casas de software y hardware ganan dinero con las empresas y las industrias y no con la educación, y el mercado de la educación especial es aun más reducido. Los centros SEMERC no pueden hacer frente a las demandas actuales de ayuda y la cantidad de escuelas especiales por cada área es inmensa. La escasez de fondos y de experiencia retrasa el desarrollo de hardware y software en los centros, y la falta de dinero en las escuelas hace que la adquisición de la tecnología de ordenadores sea lenta. No obstante, a pesar de esta carencia de recursos, se está llevando a cabo un trabajo apasionante.

Idea conceptual
El teclado conceptual responde al principio de la tablilla de gráficos. El bajar la resolución del tablero y añadir una cubierta permite que incluso los usuarios con falta de coordinación física puedan comunicarse eficazmente con el ordenador sin tener que dominar un teclado QWERTY. El diseño de las cubiertas puede ayudar a reducir la renuencia del nuevo usuario a acercarse al ordenador

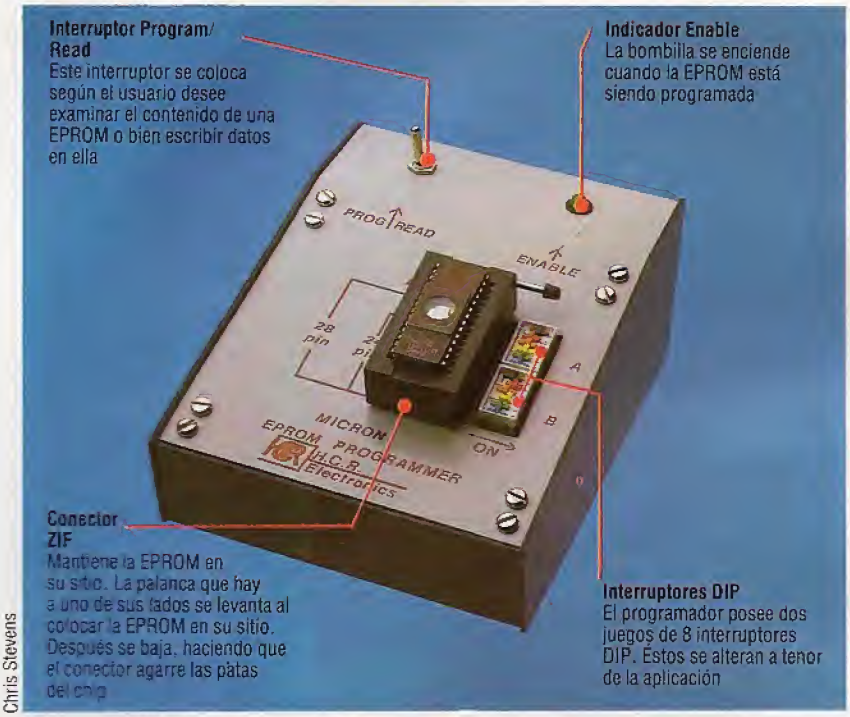
Informática y genio

Los niños dotados y de talento también tienen necesidades educativas especiales y el empleo de micros en la docencia les proporciona un medio de expresión tan interesante como lo es para los incapacitados. De hecho, muchos niños con incapacidades físicas o de percepción están muy dotados intelectualmente.

Tal como sucede con los incapacitados, el micro les proporciona a los superdotados un medio de ganar control sobre el entorno y superar las barreras naturales o impuestas por el hombre hacia el crecimiento individual y la autorrealización. Los ordenadores le ofrecen al niño superdotado:

- La oportunidad de desarrollar aptitudes e intereses a su propio ritmo;
- Formas nuevas y creativas de enfocar la resolución de problemas;
- Un medio para comunicarse y cooperar con otros estudiantes dotados;
- La oportunidad de practicar ejercicios reiterativos y agotadores de una forma que impide el aburrimiento y mantiene su interés;
- Un nuevo método (y campo) de descubrimiento.

Veloz y eficiente



Chris Stevens

Componentes del grabador
Un programador de EPROM permite que los usuarios tengan sus programas permanentemente en la memoria. Los programas se cargan en el ordenador y después se transfieren al programador a través de la puerta para el usuario, donde se "soplan" en el chip

En esta ocasión nos referiremos al Micron, dispositivo económico y eficaz para programar memorias EPROM

Mirándolo de cerca
El chip que hay dentro de la EPROM se puede observar fácilmente a través de la ventana de cuarzo del centro. Al examinarlo con detenimiento se podrán apreciar los delgados trozos de cable que unen las conexiones de E/S con las patillas y las matrices de que se compone la EPROM

Uno de los mayores méritos del BBC Micro es su flexibilidad. Al diseñar la máquina, Acorn optó por incluir cuatro ranuras para ROM adicionales, permitiendo que el usuario colocara unidades ROM extras para aplicaciones específicas en la máquina, y éstas se pueden "paginar" mediante la instrucción*. La inclusión de estas ranuras ha sido uno de los mayores éxitos de la máquina, dado que permite disponer de programas instalados en la placa a los que se puede acceder con mucha más rapidez que si se hubiera de cargar de cinta o disco.

En la actualidad existe a la venta una enorme selección de estas ROM de aplicaciones, abarcando una amplia gama de programas, incluyendo tratamiento de textos, bases de datos y lenguajes como el LOGO. Sin embargo, los usuarios también pueden escribir sus propios programas y transferirlos luego a una EPROM, que permite almacenarlos con carácter permanente en la máquina. Esto se consigue programando la EPROM con un dispositivo al que, como es natural, se denomina *programador de EPROM* y que popularmente se conoce como *soplador de EPROM*.

Al igual que todas las ROM, una EPROM se compone de una matriz de líneas eléctricas en columnas y filas, cada intersección de las cuales se denomina *celda*. De haber una conexión entre una línea de una columna y una de una fila, el estado lógico de ese punto será uno; de lo contrario, será cero. Cuando llega al bus de direcciones una serie de impulsos eléctricos correspondientes a una dirección, la electricidad pasará por ocho de estas líneas (estos ocho impulsos representan un byte). Según haya o no una conexión en las líneas de cruce apropiadas, la carga eléctrica pasará hasta las líneas de cruce y determinará el envío de una carga, a través del bus de datos, de vuelta al procesador. De este modo, los datos retenidos en una dirección se transfieren a la CPU.

Los estados lógicos en la matriz de memoria de una EPROM suelen estar establecidos en uno: cada dirección contiene el valor &FF. Si a través de una celda pasa un gran voltaje, la conexión se rompe produciendo por consiguiente el estado lógico complementario de cero. Para programar una dirección determinada de una EPROM, hemos de establecer primero el patrón de bits necesario en las patillas de dirección, para especificar la dirección a programar. Luego enviamos el código requerido por el bus de datos. A las patillas de datos apropiadas se les aplica un impulso de 50 microsegundos a 12 V, y se programa la dirección. Incrementando la dirección y enviando otro patrón de bits a las patillas de datos, se puede programar toda la EPROM. No obstante, a diferencia de otros tipos de ROM programables, en las que este proceso es irreversible, una EPROM, tal como indica su nombre, se puede borrar y programar de nuevo varias veces.

Una EPROM se caracteriza por una pequeña ventana de cuarzo que hay en la parte superior del chip. Cuando a través de esta ventana pasa una intensa luz ultravioleta, los átomos del chip se ionizan en un nivel eléctrico superior y las uniones entre los átomos se vuelven a establecer. Ello permite que la corriente pase una vez más entre las líneas de cruce de la matriz, y el estado lógico vuelve a ser uno. La luz ultravioleta, sin embargo, no es estrictamente necesaria, puesto que el programador de EPROM puede escribir una rutina de software para restaurar todas las posiciones de memoria a &FF.

El programador Micron EPROM, de HCR Electronics, le permite al usuario programar hasta 16 K de código máquina en una EPROM. El programador es una caja pequeña, cuya característica más destacada es un conector ZIF (*Zero Insertion Force*: fuerza de inserción cero). Éste permite insertar las EPROM sin dañar a las patillas que, una vez rotas, dejan al chip inutilizado. El conector ZIF está diseñado para alojar chips de 24 o 28 patillas. Junto al conector ZIF hay dos juegos de interruptores DIP, que se manipulan según cuál de las diversas opciones se utilice. En la parte superior izquierda del programador hay un interruptor que es nece-

sario posicionar en función de si se desea programar la EPROM o leerla. Obviamente, se ha de tener cuidado cuando se utiliza el programador, asegurándose de que el interruptor esté en la posición correcta, de lo contrario la EPROM se podría volver a "soplar", perdiéndose en consecuencia el programa almacenado en el chip.

El dispositivo posee un cable plano que se conecta a la puerta para el usuario del BBC desde la fuente de alimentación eléctrica. Uno puede cargar y salvar programas en un área de buffer y luego transferirlos a la EPROM con el software activado por menú que se suministra con la máquina. El software está almacenado en cassette y, como es natural, es aconsejable que lo primero que haga el usuario sea volcarlo en una EPROM.

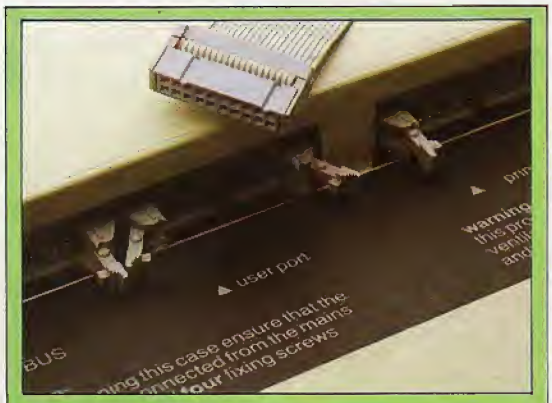
Una vez cargado, el software presenta un menú en la pantalla ofreciendo las opciones disponibles. Para copiar un programa en una EPROM, primero se debe cargar éste en la memoria utilizando la opción (L)oad (cargar). Esta instrucción tiene el efecto de colocar el programa en un área de buffer, reservada por el software EPROM, que comienza en la posición &2000. Tras haber hecho esto, el usuario selecciona la opción (P)rograma del menú principal. La pantalla visualiza entonces la lista de tipos de EPROM que se pueden programar mediante este dispositivo. Después de escoger una EPROM, la pantalla visualizará los 16 interruptores DIP y sus posiciones; los interruptores que es necesario modificar tras el programa anterior destellarán de forma intermitente. El programa solicita al usuario que mueva el interruptor de lectura/escritura a PROG. Pulsando cualquier tecla se inicializará la programación de la EPROM, visualizándose cada dirección a medida que va siendo programada.

El período de tiempo que lleva el proceso de "soplado" depende del número de direcciones de la EPROM que no vayan a ser modificadas. Ello puede consumir entre 74 segundos para un programa corto y un máximo de 14 minutos para una EPROM de 16 K. Mediante el empleo de la instrucción (V)erificar, uno puede comprobar si la EPROM se ha programado correctamente. El software calcula una suma de control, y las posibles direcciones defectuosas se listan en la pantalla. De no haber ningún error, la EPROM se puede instalar con toda seguridad en uno de los conectores para ROM del BBC Micro.

A los programas se les puede dar nombres normales. Utilizando la instrucción G, es posible agregar al comienzo del programa una rutina, asociando un nombre, al que luego se podrá llamar con el prefijo * y una instrucción CALL. Cuando se cargan en la EPROM programas en BASIC, esta instrucción se emplea junto con la opción (F)*ill* (llenar). Esta instrucción asegura que las direcciones no utilizadas por el programa en BASIC permanezcan establecidas en &FF y continúen, por tanto, siendo programables. A los programas en código máquina también se les puede dar nombres, pero éstos sólo se pueden cargar desde la dirección &2000.

Considerando lo útiles que son estos dispositivos, quizá sea sorprendente que los programadores de EPROM no hayan adquirido una mayor popularidad entre los usuarios del BBC Micro. Es probable que muchos de ellos crean que estos programadores están reservados para los amantes más "serios" de la electrónica y, por tanto, fuera del alcance de la mayoría de los usuarios de ordenadores personales. Esto no es así. Los programadores de EPROM son fáciles de usar, no requieren ningún conocimiento de electrónica, y las máquinas ofrecen numerosas ventajas. Entre éstas, una de las principales es la velocidad de acceso: se puede llamar instantáneamente a un programa a través del sistema operativo sin tener que cargarlo de disco o cinta. Además, una vez que los programas se incorporan al ordenador, las aplicaciones retenidas en la EPROM se pueden llamar desde programas en BASIC.

Por último, los usuarios disfrutan de la inmensa satisfacción de ver que los programas que ellos han escrito ocupan un lugar entre los programas incorporados del BBC Micro.



Conexiones de interface
El programador de EPROM se conecta, a través de un cable plano de 12 vías, a la puerta para el usuario del BBC Micro

EPROM BLOWER

DIMENSIONES
140×105×65 mm

INTERFACES
Conector de cable plano que se enchufa en la puerta para el usuario del BBC Micro

DOCUMENTACION
Tres hojas fotocopiadas que explican cada una de las nueve instrucciones de que dispone la máquina. Aunque la máquina es fácil de utilizar, el tono general de la documentación es un tanto elevado para el principiante

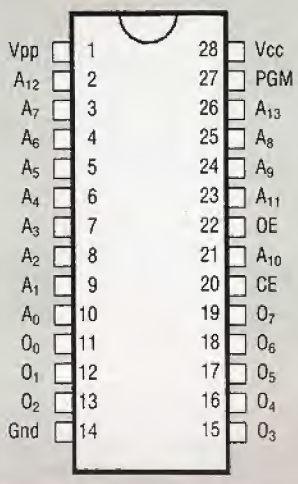
VENTAJAS
La máquina es fácil de usar. El software activado por menú no requiere por parte del usuario ningún conocimiento de la mecánica de la máquina para poder operarla

DESVENTAJAS
Poca explicación acerca de cómo podría desarrollar el usuario un uso de la máquina más allá de seguir las instrucciones del software. No es fácil transferir a EPROM los programas en código máquina que no se puedan volver a compilar para hacer que empiecen en &2000

Precisiones

Nombres de las patillas	
A_0-A_{13}	Direcciones
CE	Chip Enable (habilitación chip)
OE	Output Enable (habilitación salida)
O_0-O_7	Salidas
PGM	Programa
Vpp Vcc	Voltajes de alimentación
Gnd	Tierra

He aquí una representación de la EPROM de 16 K TMS27128, uno de los chips que se pueden "soplar" utilizando el programador HCR. Se trata de un dispositivo de 28 patillas (si bien también acepta chips de 24). El chip posee 14 patillas de dirección que le permiten direccionar los 16 K de memoria. El TMS 27128 posee, asimismo, 8 patillas para proporcionar una salida de 8 bits



Liz Dixon

Para decidir

Ahora nos ocuparemos de las construcciones para toma de decisiones: las sentencias IF y CASE

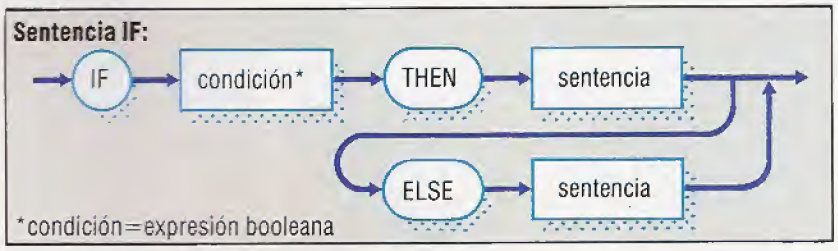
La sentencia IF del PASCAL es similar a la de la mayoría de los lenguajes. Dado que con el final de la línea no concluye una sentencia, podemos utilizar el formato libre del PASCAL para mostrar los posibles caminos a través de la estructura de la sentencia IF con indentación lógica. Éstas son dos sentencias IF:

```
IF contador=límite
  THEN
    WriteLn('No hay lugar')
  ELSE
    write('Siguiente?');
IF número>máximo THEN
  máximo:=número
```

En el segundo ejemplo, la ausencia de una cláusula ELSE implica:

```
ELSE
  {no hacer nada}
```

La alineación de las palabras reservadas THEN y ELSE (cuando está presente) ayuda a que el ojo pueda seguir el flujo del control a través de la "construcción". Se utiliza un punto y coma final para separar sentencias, tal como vemos en el ejemplo, y por lo tanto *jamás* debe aparecer antes de un ELSE. Recuerde que en PASCAL la sentencia ELSE no existe, sólo existe la sentencia IF; THEN y ELSE se emplean para delimitar la condición booleana y las sentencias de las dos cláusulas. En realidad, cada vez que utilizamos una sentencia IF estamos comprobando un valor booleano. La cláusula THEN sólo se ejecuta si la expresión evaluada es verdadera y, de seguir una cláusula ELSE, la(s) sentencia(s) en ella especificadas se llevarán a cabo si el valor booleano es falso.



El PASCAL fue el primer lenguaje que introdujo el tipo de datos conceptual conocido como escalares enumerados. Éstos son sumamente útiles, dado que nos permiten mantener una visión de alto nivel respecto a la clasificación de los datos, sin tener que traducir continuamente de forma mental los datos a códigos numéricos. (Los números son el lenguaje del ordenador, no el nuestro... a menos que estemos resolviendo un problema de naturaleza específicamente matemática.) Ya vimos cómo nos puede ser de ayuda una simple definición de constante:

```
CONST
  anchura=80;
```

El identificador de constante anchura se puede entonces utilizar a lo largo de todo el programa en referencia al número de columnas de la pantalla o la impresora. Reescribir el programa para una VDU de 40 columnas es entonces una cuestión tan sencilla como cambiar una línea de la definición del principio del programa; todos los cálculos para el formateo de la impresión, etc., se alterarán, en consecuencia, automáticamente. De este modo, se podría entonces definir una escala completa de constantes.

Si estuviéramos empleando gráficos en color, los colores disponibles se asociarían a una escala adecuada de números (rojo=1, verde=2, p. ej.), ¡pero esto teóricamente se podría utilizar entonces calculando la raíz cuadrada de verde, o multiplicando azul y amarillo! Esto no sólo es ilógico, sino que también constituye una potencial fuente de error cuando estamos razonando problemas que en sí mismos no implican datos numéricos. El elemento de definición TYPE de un programa en PASCAL se puede utilizar para definir un tipo escalar conceptual totalmente nuevo, simplemente enumerando una lista de identificadores que representen a todos los valores de constantes de la escala. Por ejemplo:

```
TYPE
  tinte=(rojo,verde,amarillo,azul,magenta,cyan);
```

Puesto que se trata de una definición (del nuevo tipo) y no de una declaración, la sintaxis emplea el signo de igualdad para definir el identificador del tipo (tinte) para referirse a los valores ordenados del tipo encerrado entre paréntesis. De forma análoga, el tipo predefinido "entero" alude a todos los números enteros disponibles en la implementación del PASCAL.

Como sucede siempre en PASCAL, los sucesivos identificadores de una lista (en este caso, valores de color) se separan entre sí mediante comas. Estos valores, por supuesto, están asociados internamente a valores enteros, con la numeración ordinal comenzando por cero. Esta representación numérica la organiza el compilador de forma automática, y es muy similar a los códigos implícitos para el juego de caracteres del ordenador. Así como el carácter A posee un código ASCII de 65, cada valor de tinte tendrá un número ordinal que podemos obtener mediante la función escalar ord. Por lo tanto, en este ejemplo, ord (rojo) es 0 y ord (cyan) sería 5. Ahora, habiendo definido este nuevo tipo escalar, podríamos declarar una variable en la forma habitual:

```
VAR
  color : tinte;
```

Esto declara un identificador (color) como un elemento de datos del tipo tinte, tal como la declaración:

```
VAR
  letra : char;
```

indica la naturaleza de carácter del objeto de datos denominado letra. Las únicas operaciones definidas sobre tipos de enumeración son las comparaciones de relación y el empleo de funciones escalares. Por ejemplo, podríamos escribir:

```
if color<cyan then
  color:=succ(color)
```

¡Recuerde que pred(rojo) y succ(cyan) no existen! La variable color es incompatible con variables de cualquier otro tipo, escalares u otras. Ello significa que ya no podemos efectuar funciones ilegales, como extraer raíces cuadradas o decir:

```
color:=color+1
```

Esto conduce a una restricción evidente. Los caracteres y los números se pueden utilizar como parámetros para sentencias write y WriteLn, pero

```
WriteLn(color)
```

sería ilegal. Ello se debe a que los valores del tipo son puramente conceptuales y si deseamos imprimir sus nombres, debemos asociar los valores de color a series de caracteres. Ésta es una aplicación ideal para la otra construcción de opciones del PASCAL, la sentencia CASE.

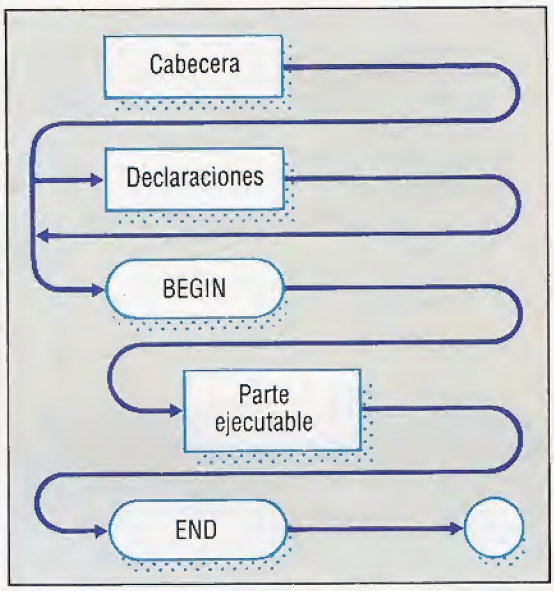
Hemos visto lo familiar que nos resulta la sentencia IF del PASCAL y los beneficios que suponen las convenciones de libre formato del PASCAL desde el punto de vista de la legibilidad. Hay ocasiones, sin embargo, en las que se han de tomar decisiones de múltiples opciones que requerirían algo como:

```
IF N=1
  THEN
    write ('primero')
  ELSE
    IF N=2
      THEN
        write ('segundo')
      ELSE
        IF N=3
          THEN
            write ('tercero')
          ELSE
            write ('mayor de tres')
```

Siempre que la sentencia a ejecutar dependa de que el valor de una variable escalar simple esté comprendido en una gama limitada de valores podemos utilizar la sentencia CASE por razones de comodidad. En este caso:

```
CASE N OF
  1              :write ('primero');
  2              :write ('segundo');
  3              :write ('tercero');
  4,5,6,
  7,8,9          :write ('mayor de tres')
END  {CASE}
```

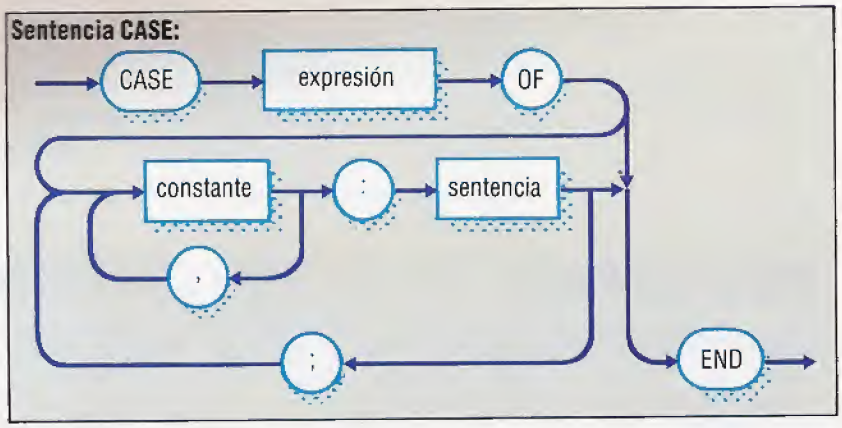
Observe que esto sólo es satisfactorio para valores de N comprendidos en la escala de "etiquetas de CASE" especificada en el "cuerpo" de la sentencia



Esqueleto de un programa
Éste es el diagrama sintáctico simplificado de un programa en PASCAL, que muestra los principales elementos. Observe que las palabras reservadas del PASCAL aparecen en mayúsculas. Ésta es una convención que se ha desarrollado para diferenciarlas de las palabras no reservadas. La mayoría de las versiones de PASCAL, sin embargo, aceptarán palabras reservadas en minúsculas

CASE: entre 1 y 9, en este ejemplo. Todos los valores que pudiera tener N en tiempo de ejecución se *deben* especificar individualmente y sería ilegal, por ejemplo, que se entrara N=0.

Muchos compiladores de PASCAL poseen la palabra reservada adicional OTHERWISE u OTHERS, que se puede emplear para especificar una acción por defecto. Remítase a su manual para la sintaxis necesaria en este caso.



Liz Dixon

Ésta es la única sentencia del PASCAL que utiliza END para delimitar una estructura que no se haya entrado mediante un BEGIN, de modo que es una práctica común, y de buen estilo, calificar cada END de una sentencia CASE tal como se indica.

La construcción trabaja evaluando la expresión escalar delimitada por las palabras reservadas CASE y OF. Un nombre de variable simple es una expresión mínima que no requiere cálculo. El valor obtenido se compara entonces con cada una de las constantes especificadas en las listas de etiquetas del CASE, y cuando se halla una pareja se ejecuta la sentencia que sigue a los dos puntos, y *sólo* esa sentencia. El flujo de control, por consiguiente, no se desvía, de modo que se mantiene la integridad estructural de la construcción. Si fuera necesario llevar a cabo varias operaciones, se podría utilizar, por supuesto, una sentencia compuesta encerrada entre un par BEGIN/END.

En ciertas circunstancias, algunos valores po-

drían no requerir ninguna acción, en cuyo caso se puede utilizar la sentencia más simple de todas las sentencias del PASCAL. Ésta es la sentencia "nula", que significa "no hacer nada" y ¡carece absolutamente de sintaxis!

He aquí un ejemplo:

```
CASE N MOD 4 OF
  0 :{no hacer nada};
  1,3: begin
       write (N MOD 4 : 1,'cuarto');
       if N MOD 4>1 then
         write ('s')
     end;
  2 :write ('y medio')
END {CASE}
```

Observe que sigue siendo necesario un punto y coma para separar esta sentencia inexistente de lo que sigue a continuación. La sentencia de la última etiqueta no requiere ninguno porque va seguida de una palabra reservada (END), no por otra etiqueta o sentencia.

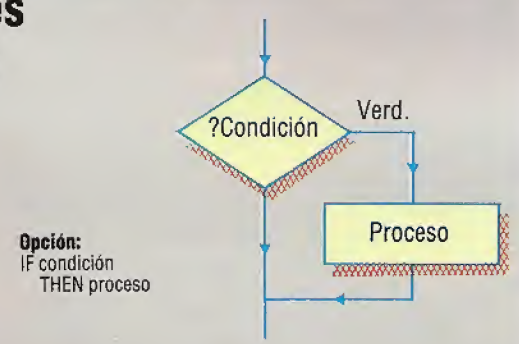
El empleo del operador MOD en la expresión asegura que el valor debe estar en la escala entre 0 y 3. MOD da el resto de una división entre enteros, como en el BASIC BBC y otros.

En el próximo capítulo de la serie examinaremos estos y todos los otros operadores del PASCAL, así como las funciones incorporadas. Mientras tanto, he aquí la solución al problema de cómo imprimir las series de caracteres para cada valor de nuestro tipo de color y tinte:

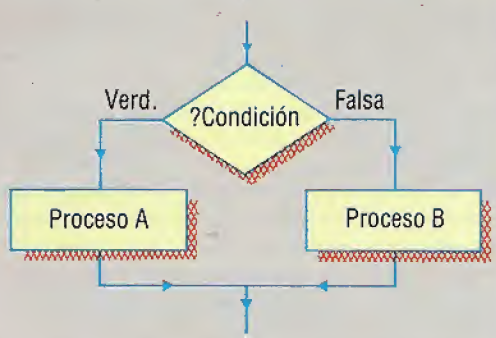
```
CASE color OF
  rojo        :write ('Rojo');
  verde       :write ('Verde');
  amarillo    :write ('Amarillo');
  azul        :write ('Azul');
  magenta     :write ('Magenta');
  cyan        :write ('Cyan');
END {CASE}
```

Construcciones de decisiones

La construcción IF...THEN ejecutará el "proceso" si la condición resulta verdadera. Si es falsa, el programa sigue en la siguiente sentencia. Mientras que la construcción IF...THEN presenta una "opción", IF...THEN...ELSE presenta una "elección". Según el resultado de la comparación realizada al comienzo de la construcción, se ejecutará uno de los dos procesos



Opción:
IF condición
THEN proceso



Elección:
IF condición
THEN proceso A
ELSE proceso B

Sentencias mensuales

Este programa, que simplemente calcula la cantidad de días entre una fecha entrada por el usuario y el final del mes, ofrece algunos útiles ejemplos de los principios de la programación en PASCAL. La sentencia CASE se emplea para asociar datos de entrada numéricos a los tipos enumerados del PASCAL, utilizándolos para el proceso interno y luego volviéndolos a asociar a datos de salida en serie. Observe el empleo de la variable *char* ficticia (símbolo) para leer los caracteres delimitadores que separan los números entrados

```
PROGRAM          Fecha        (input, output);

CONST
  PuntoFinal='.';

TYPE
     Calendario=(Ene, Feb, Mar, Abr, May,
          Jun, Jul, Ago, Sep, Oct, Nov, Dic);

VAR
     NombreMes        :Calendario;
     dia,
     mes,
     año,
     Faltan           :integer;
     simbolo          :char;
     AñoBisiesto      :boolean;

BEGIN
  WriteLn ('Entre la fecha en la forma :');
  WriteLn ('DD/MM/AA' : 40 );
  WriteLn;
  write ('Fecha ?');
  read (dia,simbolo,mes,simbolo,año);
  AñoBisiesto:=año MOD 4=0;
                    (ignorando siglos)

  IF (mes>0)AND(mes<=12)
    THEN
      CASE mes OF

        1          :NombreMes :=Ene;
        2          :NombreMes :=Feb;
        3          :NombreMes :=Mar;
        4          :NombreMes :=Abr;
        5          :NombreMes :=May;
        6          :NombreMes :=Jun;
        7          :NombreMes :=Jul;
        8          :NombreMes :=Ago;
        9          :NombreMes :=Sep;
        10         :NombreMes :=Oct;
        11         :NombreMes :=Nov;
        12         :NombreMes :=Dic;

      END          (CASE)
    ELSE
      BEGIN
        WriteLn          ('Eh ?');
        WriteLn          (' - el programa ',
                          'casi se ROMPE !')
      END;               (NombreMes no está inicializado)

    CASE NombreMes OF

    Ene, Mar,
    May, Jul         : Faltan :=31 - dia;

    Abr, Jun,
    Sep, Nov         : Faltan :=30 - dia;

    Feb              : IF AñoBisiesto
                          THEN
                            Faltan :=29 - dia
                          ELSE
                            Faltan :=28 - dia

  END; (CASE)

  WriteLn;
  write ('Faltan ', Faltan : 1,
         'dias ');

  CASE NombreMes OF

          Ene                :write ('Enero');
          Feb                :write ('Febrero');
          Mar                :write ('Marzo');
          Abr                :write ('Abril');
          May                :write ('Mayo');
          Jun                :write ('Junio');
          Jul                :write ('Julio');
          Ago                :write ('Agosto');
          Sep                :write ('Septiembre');
          Oct                :write ('Octubre');
          Nov                :write ('Noviembre');
          Dic                :write ('Diciembre');

  END; (CASE)

  WriteLn (PuntoFinal)

END.
```

Otros juicios

Nuestra comentarista Virginia Rizla hace la valoración crítica de dos programas para el Sinclair Spectrum: uno es de tema deportivo; el otro, de índole "científica"

Daley's decathlon

Ocean

Al haber dejado de lado hace bastante tiempo todo lo que se asemejara siquiera remotamente al ejercicio físico, la perspectiva de tener que jugar al *Daley's decathlon* (El decatlón de Daley) no me causó demasiada alegría, antes bien una acusada sensación de indiferencia.

El objetivo del juego es, naturalmente, competir en las diez pruebas del decathlon (100 m lisos, salto de longitud, lanzamiento de peso, salto de altura, 400 m lisos, 110 m vallas, lanzamiento del disco, salto con pértiga, lanzamiento de jabalina y 1 500 m lisos) y conseguir la máxima cantidad de puntos en cada una. En las pruebas de carrera, el movimiento se controla alternando dos teclas o bien maniobrando la palanca de mando; tras pulsar otra tecla, me encontré saltando enormes distancias y esquivando objetos con notable habilidad y gracia.

Aunque vacilante en la mayor parte del juego, me sentí muy alentada por la clamorosa multitud al conseguir 128 m en el salto de longitud (¡atención a eso, Carl Lewis!). Y no me sentí alienada en lo más mínimo por los convencionales gráficos, por demás realistas, y movimientos.

En resumen, les aconsejo a todos que no boicoteen esta competición.

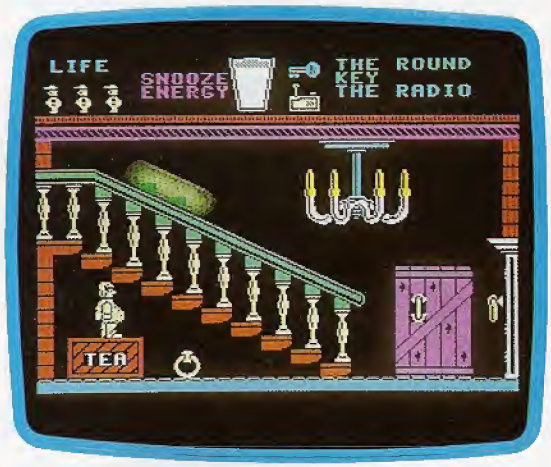
Pyjarama

Micro-Gen

Wally tiene una pesadilla y procede a refutar varias de las tesis de Jung sobre el sonambulismo. (Si realmente está inconsciente, parece un poco extraño que el objetivo del juego sea hallar el botón de su reloj despertador.)

Sin embargo, *Pyjarama* se nos revela como un juego muy entretenido y divertido y no resulta en absoluto irreverente con el psicoanálisis ni las más modernas teorías respecto a la interpretación de los sueños y a las fuerzas que gobiernan nuestro inconsciente.

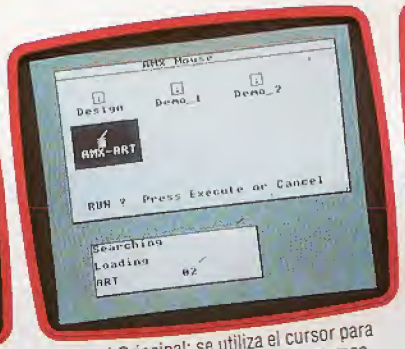
Varias pantallas representan las habitaciones de la morada de Wally, dibujadas de forma maravillosa y con brillantes colores. Moviéndose a izquierda y derecha y saltando sobre el mobiliario, de dudoso gusto, Wally intenta atravesar las diversas puertas, ya sea saltando para alcanzar el picaporte o gracias a poseer cierto "objeto". Dado que su energía vital es limitada, es preferible evitar los gráficos flotantes y las manos que tienden a aparecer por debajo del suelo.



Gran parte del entretenimiento consiste en aprender a jugar al juego, tras lo cual sigue teniendo atractivo visual y, lo que es más importante, interés lúdico.

Arte iconográfico

editor de iconos le permite al usuario crear o aptar el juego de iconos



El Menú Principal: se utiliza el cursor para seleccionar uno de los cuatro programas

Para crear una imagen: primero se dibuja el contorno

Hay un menú para seleccionar las diferentes líneas disponibles

Analizamos el paquete de ratón AMX, que incluye el programa "AMX Art", creado para el BBC Micro

AMX Mouse
El ratón AMX forma parte de un paquete completo que incluye el ratón propiamente dicho, un chip de ROM que contiene el software activador, software de aplicaciones en disco o cassette, un manual para el usuario y un manual de aplicaciones del *AMX Art*. En cuanto a ejecución, el paquete es muy similar al utilizado en el Apple Macintosh, pero los usuarios del BBC lo tienen a su alcance por una fracción de su precio

El paquete de ratón AMX, producido por Advanced Memory Systems para el BBC Micro, se compone del ratón propiamente dicho, junto con un cable para conectarlo a la puerta para el usuario del micro, un chip de ROM, dos manuales y software en disco y cassette. Si bien se suministran tanto cassette como disco, la propia AMS admite que el AMX está claramente dirigido a quienes poseen unidad de disco.

El AMX, un ratón estándar fabricado en Japón, está moldeado en plástico negro con una bola metálica en su base para la detección del movimiento.

Ian McKinnell

El ratón cabe con comodidad en la mano, dejando los tres primeros dedos descansando sobre los tres botones para selección de acción: Move, Execute y Cancel. La decisión de incorporar una bola metálica es poco afortunada, dado que tiende a resbalar muchísimo sobre superficies tales como madera encerada y superficies de escritorio laminadas. Por consiguiente, es una buena idea colocar un trozo de papel sobre cualquier superficie resbaladiza.

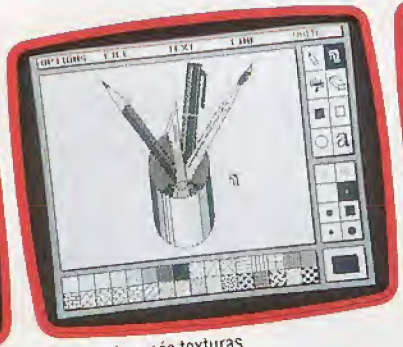
En el interior del ratón, la bola está apoyada en dos rodillos; al extremo de cada uno hay un disco con muescas que pasa a través de una célula fotoeléctrica. El disco corta un haz de luz, creando reflejos intermitentes que son captados por la célula, y éstos se transmiten al ordenador como instrucciones de dirección. Por consiguiente, el ordenador conoce con exactitud en todo momento la posición, el movimiento y la velocidad del ratón.

El software residente en la ROM lateral corresponde a la comunicación con el ordenador. Además de proporcionar rutinas "en bruto" para su empleo con software comercial, esta ROM facilita varias rutinas de fácil uso que permiten utilizar el ratón con programas de elaboración casera. Éstas incluyen, entre otras, una rutina para proporcionar una posición actualizada de las coordenadas en pantalla del ratón, otra para leer los botones de selección y una tercera para dar animación a los iconos en la pantalla. Haciendo uso de estas rutinas (todas ellas seleccionadas mediante el empleo de la estructura de instrucción * del sistema operativo del BBC Micro), el usuario puede escribir rápida y fácilmente programas en BASIC que saquen el máximo partido del control del ratón.

También se proporcionan rutinas que permiten el empleo del ratón en paquetes de software comercial ya existentes. El movimiento del ratón se puede configurar de modo que sustituya el uso de las teclas del cursor, y los tres botones de selección para que actúen como tres teclas cualesquiera del teclado, incluyendo las teclas Shift y CTRL. De esta forma es posible, por ejemplo, utilizar el ratón para suprimir, trasladar y copiar secciones de texto en el procesador de textos *View* de Acornsoft.

Sin embargo, la bondad del dispositivo se hace verdaderamente ostensible cuando se utiliza software de aplicaciones escrito especialmente para el

Los contornos se rellenan con varios patrones diferentes...



Y se añaden más texturas

Se agregan palabras y letras pasando a la modalidad de texto y eligiendo el tipo

Mediante la modalidad de mano alzada se los toques finales

control por ratón. El paquete AMX incluye dos ejemplos de este tipo de software, que demuestran ampliamente la sencillez y naturalidad del empleo del ratón. El primero es un programa en BASIC para diseñar iconos con el objeto de incorporarlos en los programas creados por el usuario. Los iconos son símbolos gráficos que se pueden visualizar en un programa para indicar las opciones del menú o incluso como sustituto del cursor para indicar la selección efectuada.

El paquete de diseño *Icon* presenta una cuadrícula de 16×16 en la cual usted puede diseñar su icono rellenando los cuadrados deseados: todo ello mediante el ratón. En realidad, con este programa la única ocasión en la que tendrá necesidad de utilizar el teclado será para entrar el nombre de archivo bajo el cual se ha de guardar un grupo de iconos, o el nombre del archivo de un conjunto ya existente que se deba cargar. Debido al enorme control que se ejerce sobre el ratón, éste es sumamente fácil de usar, y diseñar iconos complejos y realistas es una labor rápida y sencilla. No obstante, el paquete *AMX Art*, que también se suministra, ofrece una mejor demostración del potencial del ratón. Éste es un programa para diseño gráfico con características completas y controlado totalmente mediante el ratón. Por su sencillez de uso es comparable al paquete *Macpaint* para el Macintosh y, de hecho, la pantalla del AMX guarda un notable parecido con la creación de Apple.

El programa de arte

El dibujo se efectúa en la zona central de la pantalla; en el programa se utiliza una visualización en Mode 4, de 320 por 256, en dos colores. Sobre el lado derecho de la pantalla hay dos menús de iconos. El menú superior se utilizar para seleccionar diversas acciones de dibujo, incluyendo trazado de líneas, borrado, efectos de pintura con aerosol y rellenado de superficies, que se representan en la pantalla mediante formas fácilmente reconocibles. Una acción se puede seleccionar desplazando el cursor encima del icono en cuestión y pulsando luego uno de los botones de selección, lo cual hace que el cursor cambie de inmediato a la forma del icono seleccionado. Entonces se puede construir la imagen desplazando el cursor a través de la zona de dibujo; mediante el empleo de los botones de selección uno puede llevar a cabo cualquiera de las funciones de dibujo. El excelente manual que se entrega con el *AMX Art* apenas si resulta necesario, debido en parte al uso tan directo del programa y también a que el movimiento del cursor reproduce con gran exactitud el movimiento del ratón.

El menú inferior del lado derecho de la pantalla selecciona la anchura y forma de las líneas de la misma manera. En la parte inferior de la pantalla hay un tercer menú para seleccionar los patrones de sombreado para rellenar las formas, que van desde el negro hasta complejos patrones de enrejado. Puesto que la visualización es en sólo dos colores (por cuestiones de limitación de memoria), se ofrecen estos patrones para diferenciar distintas áreas mediante diversos tipos de sombreado.

Otras opciones, tales como guardar (SAVE) o cargar (LOAD) imágenes, instrucciones del sistema operativo y activación del programa incorporado para vuelcos de pantalla, se pueden seleccionar mediante los menús de la parte superior de la pantalla. Sólo se visualizan los títulos de estos menús, y para revelar el menú completo sobre una sección de su imagen debe posicionarse el cursor sobre uno de los títulos y pulsar uno de los botones de selección. Tras seleccionar sus opciones con el ratón, el menú desaparece dejando la imagen en su forma original.

Desde la primera vez que utilice este programa podrá realizar composiciones complejas, exactas y artísticamente innovadoras con sencillez y rapidez. Tras emplear este paquete durante un breve período, se convierte en un medio tan cómodo como el lápiz y el papel, con la excepción de que el *AMX Art* nunca pierde su filo ni se mancha, y que se puede reproducir tantas veces como se desee.

El programa *Art* ya es de por sí una razón de peso para adquirir el paquete de ratón, pero AMX tiene la intención de ampliar esta primera iniciativa para convertirla en un sistema completo basado en ratón para el BBC Micro. Hay en fase de preparación un gestor de sobremesa de operatoria similar al utilizado en los ordenadores Macintosh y Lisa. También existen planes para una base de datos activada por ratón y más mejoras para el programa *AMX Art*, al cual se le incorporará color y una facilidad de *zoom*.

El ratón AMX y el programa *AMX Art*, así como los otros programas de ratón e iconos que están en preparación, parecen estar llamados a lavarle la cara al BBC Micro, al menos para el usuario que desee ampliar las capacidades de la máquina. Y, por su sencillez de uso, estos programas han establecido un nuevo estándar para los ordenadores personales.

AMX MOUSE

DIMENSIONES

85×65×35 mm

INTERFACES

Un cable a conectar en la puerta para el usuario del BBC Micro

DOCUMENTACION

Dos manuales de fácil comprensión y muy informativos, si bien apenas son necesarios para el paquete *Art*, gracias a la gran amabilidad del paquete hacia el usuario

VENTAJAS

Fácil de operar, manuales excelentes, el paquete *Art* posee muchas capacidades y está bien adaptado al sistema operativo del BBC

DESVENTAJAS

Aparte del paquete *Art*, el software es algo limitado. La bola de metal patina sobre las superficies muy pulidas

¡Todos a bordo!

Iniciamos la creación de un juego basado en una expedición comercial al Nuevo Mundo durante el siglo XVI

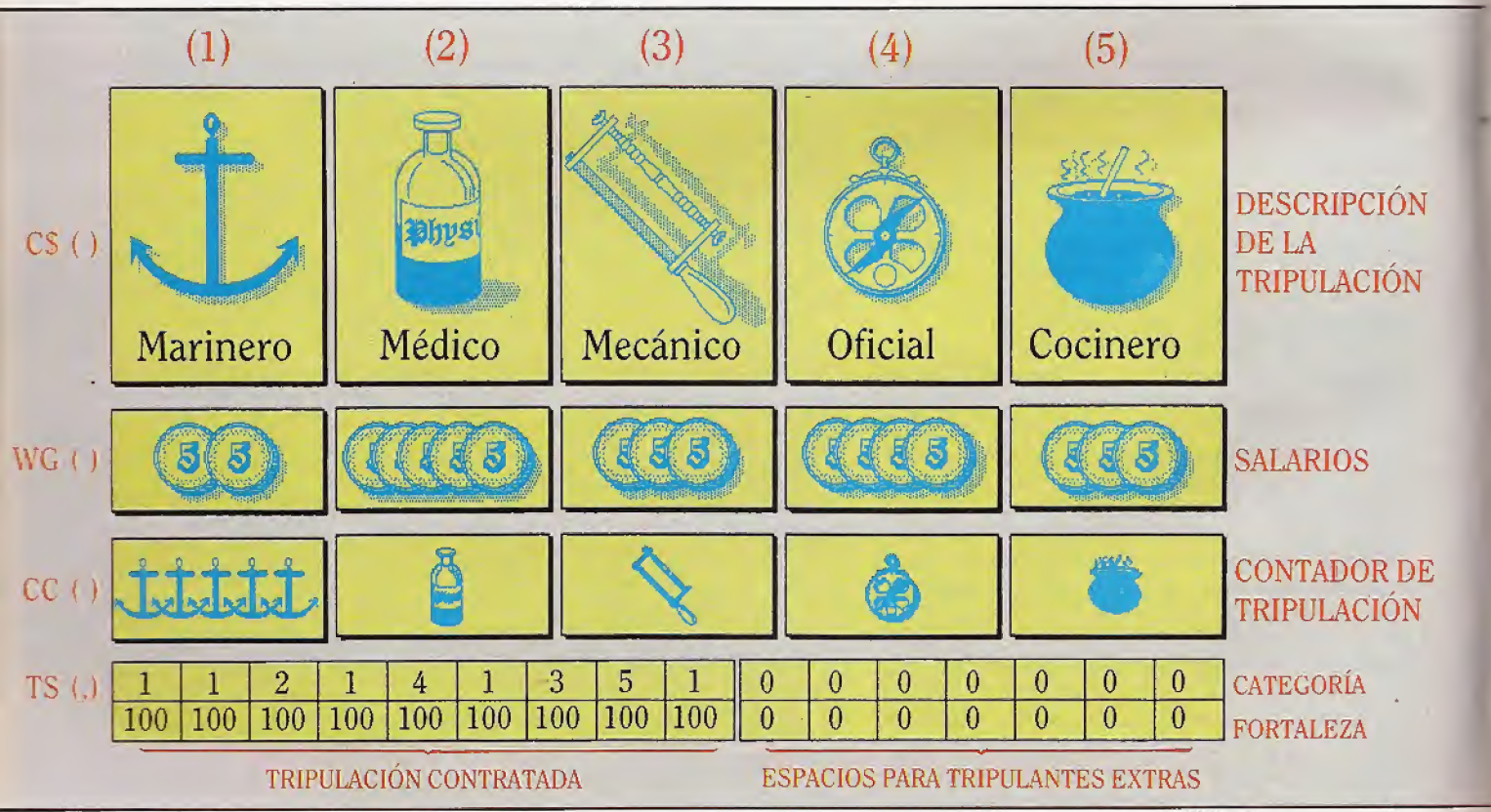
En este juego de simulación usted retrocede en el tiempo y asume el papel de un comerciante del s. XVI que decide organizar una expedición al Nuevo Mundo. Debe planificar el viaje, realizar la travesía y, por último, efectuar transacciones comerciales (de forma rentable, es de esperar) con los nativos.

Al comenzar el juego, el jugador es propietario de un barco mercante y posee 2 000 piezas de oro para gastar. Su primera tarea consiste en contratar una tripulación, ofreciéndole a cada integrante de ella un salario semanal durante la expedición; el monto de los pagos dependerá de la capacidad que posea cada miembro de la tripulación. Ésta debe cubrir una serie de aptitudes, sin que por ello usted se vea obligado a hacer frente a un presupuesto excesivo para la paga de salarios. El navío tiene capacidad para una tripulación compuesta por dieciséis personas; cuanto más numerosa sea, más cara resultará, y consumirá más cantidades de comida y bebida, dejando un menor margen de dinero para las actividades mercantiles. No obstante, si la tripulación fuera demasiado reducida no sería capaz de afrontar algunas eventualidades que pueden producirse durante el viaje, como una epidemia de escorbuto y ataques de piratas, y la travesía no se podría efectuar de forma eficaz, por lo cual la expedición duraría más tiempo y costaría más oro.

El primer módulo trata de la inicialización de variables y matrices que se utilizarán durante el juego y en la contratación de los miembros de la tripulación. La primera sección del módulo DIMensiona las matrices que almacenan importante información acerca de los miembros de la tripulación. C$() se utiliza para retener descripciones de las categorías de tripulantes. Dado que hay cinco categorías de tripulación, la matriz se DIMensiona para tener cinco elementos en la línea 16. Las descripciones se le asignan luego de forma individual a cada elemento de la matriz. Es mucho más conveniente (y ahorra memoria) referenciar las descripciones de la tripulación por su número de elemento en esta matriz. Este número de elemento se conoce algunas veces como la dirección de la matriz, dado que da la posición que ocupa la descripción en la matriz.

El estado de la tripulación varía a lo largo del juego, aumentando o disminuyendo su fortaleza a tenor de las circunstancias. Para llevar un registro de la constitución de la tripulación y de la condición física de cada uno de sus miembros, se utiliza una matriz bidimensional, TS(,). La matriz se DIMensiona en 16 columnas y dos filas. Cada columna representa a un tripulante; la fila superior contiene la categoría que le corresponde y la inferior el coeficiente de fortaleza.

Patrulla de reclutamiento
Se utilizan cuatro matrices para retener los datos relativos a la tripulación contratada para el viaje. CS(), WG() y CC() retienen las características propias de cada categoría de tripulante: descripción, salario y número. TS(,) retiene la composición actual de la tripulación seleccionada para la travesía, hasta un máximo de 16 miembros. Cada una de las categorías de tripulantes puede ser referenciada en función de un número en vez de un nombre, correspondiendo el primero a una dirección de las tres matrices de características de la tripulación



Ian McKinnell

Inicialmente, cada tripulante posee un coeficiente 100 de fortaleza.

El salario de cada categoría de tripulante también se puede representar en una matriz de cinco elementos, WG(). Por consiguiente, si deseamos saber el salario de un médico, categoría de tripulante 2, podemos consultarlo en esta matriz como WG(2). Los marineros reciben 10 piezas de oro por semana, los médicos cobran 25, los oficiales 20 piezas de oro y los cocineros, 15 cada uno. Para llevar el registro de cuánto dinero posee el jugador, en la línea 12 se inicializa una variable, MO, en 2000, que irá decreciendo a medida que se abonen las facturas salariales semanales y se inicien las transacciones comerciales.

Más adelante el programa necesitará saber cuántos tripulantes de cada categoría se han contratado, para poder calcular la eficacia de la tripulación para afrontar las contingencias que puedan producirse. La constitución de la tripulación se podrá determinar, cada vez que se requiera, explorando la fila superior de la matriz TS; pero es más rápido y más fácil preparar otra matriz, CC(), para retener la cantidad de miembros contratados de cada categoría. Nuevamente, esta matriz sólo requiere cinco elementos, y se DIMensiona en la línea 18.

Información preliminar

Tras establecer estas matrices que retienen datos sobre la tripulación, el módulo del programa imprime (PRINT) información relativa a la duración aproximada de la travesía y a la cantidad de dinero que se debe separar para adquirir provisiones. Esta información no se imprime en su totalidad, sino que se emplea una rutina especial para hacer que las letras vayan apareciendo lentamente en la pantalla. Esta subrutina, de la línea 9100, acepta la frase a imprimir en la variable S$ e imprime en la pantalla un carácter por vez, insertando una breve demora para conseguir un efecto de teletipo. Una segunda subrutina, en la línea 9200, inserta pausas más prolongadas, cuando así se requiera, entre la aparición de una línea de texto y la línea siguiente.

La subrutina de la línea 1000 le permite seleccionar su tripulación. Ello se realiza seleccionando una categoría de tripulante entre 1 y 5, o pulsando F para finalizar. Las entradas válidas se almacenan en TS(,) en la línea 1156, llevando CN la cuenta del número de tripulantes contratados hasta el momento. A medida que se va contratando a cada miembro de la tripulación, se visualiza la factura de salarios total por semana, WT, que se va calculando sumándole el elemento correspondiente a la matriz WG() al valor anterior de WT. Esto sucede en la línea 1158. A cada etapa, la sección del programa entre las líneas 1160 y 1190 imprime la composición en curso de la tripulación. La línea 1185 verifica si el contador para una categoría determinada es mayor de uno o cero. De ser así, se añade una S al final de la descripción de las categorías de tripulantes.

En varias etapas del programa se le solicita al jugador que pulse una tecla antes de seguir adelante. La variable K$ se utiliza para retener el mensaje "PULSE CUALQUIER TECLA PARA CONTINUAR" y se transfiere a S$ para su impresión cada vez que se requiera el mensaje. Al retornar de la subrutina de contratación de tripulación, termina el primer módulo.

Módulo Uno

```
 9 K$="  PULSE CUALQUIER TECLA PARA CONTINUAR"
10 DIM TS(16,2):REM TIPO/FORTALEZA DE LA TRIPULACION
11 CN=0:REM NUMERO DE TRIPULANTES
12 MO=2000: REM DINERO INICIAL
13 DIMWG(5):WG(1)=10:WG(2)=25:WG(3)=15:WG(4)=20:WG(5)=15:REM SALARIOS
14 WT=0:REM FACTURA SALARIAL SEMANAL
15 CM=16:REM TRIPULACION MAX
16 DIMC$(5):C$(1)="MARINERO":C$(2)="MEDICO":C$(3)="MECANICO"
17 C$(4)="OFICIAL":C$(5)="COCINERO"
18 DIMCC(5):REM CONTADOR DE CADA TIPO DE TRIPULACION
80 PRINTCHR$(147):S$="  JUEGO MERCANTIL NUEVO MUNDO*":GOSUB9100:PRINT
81 GOSUB9200
82 S$="ERES EL CAPITAN DE UN BARCO*":GOSUB9100:PRINT
83 S$="QUE SE DIRIGE AL NUEVO MUNDO. *SE*":GOSUB9100:PRINT
84 S$="CALCULA QUE EL VIAJE DURARA OCHO*":GOSUB9100:PRINT
85 S$="SEMANAS, PERO PODRIA DURAR MAS. DEBES*":GOSUB9100:PRINT
86 S$="CONTRATAR UNA TRIPULACION, PAGARLES, COMPRAR*":GOSUB9100:PRINT
87 S$="PROVISIONES, EQUIPO Y MERCANCIAS*":GOSUB9100:PRINT
88 S$="PARA COMERCIAR. DISPONES DE 2000 PIEZAS*":GOSUB9100:PRINT
89 S$="DE ORO PARA GASTOS. *":GOSUB9100:PRINT:GOSUB9200
90 PRINT:S$="        BUENA SUERTE!*:GOSUB9100:GOSUB9200:PRINT
91 S$=K$:GOSUB9100
94 GETP$:IFP$=""THEN94
95 GOSUB9200
500 GOSUB1000
999 END
1000 PRINTCHR$(147):PRINT"ETAPA 1 – CONTRATAR LA TRIPULACION"
1010 PRINT"                    ----------------------------------------
1012 PRINT
1015 GOSUB9200
1020 PRINT:PRINT"CATEGORIAS DE TRIPULANTES DISPONIBLES:"
1025 GOSUB9200
1030 PRINT
1040 PRINT"CATEGORIA  DESCRIPCION     SALARIO SEMANAL"
1050 PRINT"  --------    ----------      ---------------"
1060 PRINT"     1     MARINERO      10 PIEZAS DE ORO"
1070 PRINT"     2     MEDICO        25 PIEZAS DE ORO"
1080 PRINT"     3     MECANICO      15 PIEZAS DE ORO"
1090 PRINT"     4     OFICIAL       20 PIEZAS DE ORO"
1100 PRINT"     5     COCINERO      15 PIEZAS DE ORO"
1105 GOSUB9200
1110 PRINT:PRINT:PRINT
1120 S$="ENTRE LA CATEGORIA DE TRIPULANTE REQUERIDA (1–5)*":GOSUB9100
1122 S$="0  'F' PARA FINALIZAR LA CONTRATACION*":GOSUB9100:PRINT:INPUTP$
1125 CT=VAL(P$)
1128 IFLEFT$(P$,1)="F"THENPRINT:PRINT"FIN DE LA CONTRATACION DE
     TRIPULACION.":GOSUB9200:GOTO1310
1130 IFCT>0ANDCT<6THEN1150
1139 PRINT:PRINT
1140 PRINTP$;:S$="  NO ES UNA CATEGORIA DE TRIPULANTE*":GOSUB9100
1142 GOSUB9200
1145 S$="INTENTELO NUEVAMENTE, POR FAVOR"
1146 GOSUB9100
1147 GOTO1300
1150 PRINT:PRINT
1155 CN=CN+1:REM TRIPULACION CONTRATADA HASTA AHORA
1156 TS(CN,1)=CT:REM CATEGORIA DE TRIPULANTE
1157 TS(CN,2)=100:REM FORTALEZA INICIAL
1158 WT=WT+WG(CT):REM SALARIOS TOTALES
1159 CC(CT)=CC(CT)+1:REM CONTADOR CATEGORIAS DE TRIPULACION
1160 S$="TRIPULACION HASTA EL MOMENTO:"
1170 FORT=1TO5
1180 PRINTSS;CC(T);"  ";C$(T);
1185 IFCC(T)>1ORCC(T)=0THENPRINT"S";:GOTO1189
1186 PRINT"  "
1189 S$=""
1190 NEXT
1195 PRINT:PRINT"FACTURA SALARIAL SEMANAL TOTAL ";WT
1200 IFCN=CM-1 THENPRINT:S$="SOLO UN TRIPULANTE MAS*":GOSUB9100:GOTO1295
1202 IFCN=CMTHENPRINT:S$="      EL BARCO YA ESTA COMPLETO!!!":GOSUB9100:GOTO1310
1295 REM
1300 GOTO1015
1310 PRINT:  S$=K$:GOSUB9100:PRINT:GOSUB9200
1320 GETP$:IFP$=""THEN1320
1999 RETURN
9100 REM IMPRESION LENTA DE S$
9110 FOR S3=1TO32
9115 IFMID$(S$,S3,1)="*"THENS3=32:GOTO9140
9120 PRINTMID$(S$,S3,1);
9130 FORS4=1TO25:NEXT
9140 NEXT:PRINT
9199 RETURN
9200 REM BUCLE DE DEMORA
9210 FORS5=1TO1000:NEXT
9299 RETURN
```

Complementos al BASIC

Spectrum: Introduzca estas modificaciones:

```
16 DIM C$(5,9):C$(1)="MARINERO":C$(2)=
   "MEDICO":C$(3)="MECANICO"
17 C$(4)="OFICIAL":C$(5)="COCINERO"
80 CLS:LET S$="JUEGO MERCANTIL
   NUEVO MUNDO*":GOSUB9100:PRINT
94 LET P$=INKEY$:IF P$="" THEN GO TO 94
1005 CLS: PRINT"ETAPA 1 – CONTRATAR
     TRIPULACION"
1128 IF P$(1 TO 1)="F"THEN PRINT
     "FINALIZADA LA CONTRATACION DE
     TRIPULACION.":GOSUB9200:GOTO1310
9115 IF S$(S3 TO S3)="*" THEN
     S3=32:GOTO9140
9120 PRINT S$(S3 TO S3);
```

BBC Micro: Introduzca estas modificaciones:

```
80 CLS:S$="JUEGO MERCANTIL NUEVO
   MUNDO*":GOSUB9100:PRINT
94 P$=GET$
1005 CLS:PRINT"ETAPA 1 –
     CONTRATACION TRIPULACION"
```

Haciendo los preparativos...
El primer módulo del juego mercantil *Nuevo Mundo* está relacionado con la contratación de la tripulación. Con este fin se establecen varias variables y el jugador puede seleccionar hasta 16 tripulantes

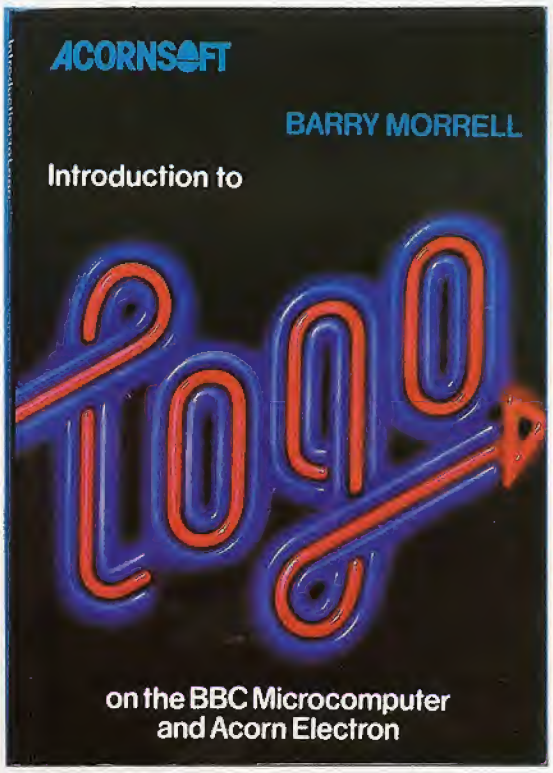
La tortuga va a la escuela

La combinación de BBC Micro y LOGO se está convirtiendo en uno de los sistemas informáticos más utilizados en las escuelas. He aquí dos paquetes escritos para esta máquina

La pantalla de gráficos es de alrededor de 1200 por 800 pasos de tortuga, dando una resolución mejor que la de los paquetes de LOGO para la mayoría de los otros micros. Están disponibles las ocho modalidades de pantalla del BBC Micro, así como las diversas opciones de dibujo (líneas de puntos, relleno triangular, etc.). Si usted utiliza Mode 7, dejará libre muchísima más memoria para aplicaciones que no requieran gráficos. Para sonido, se proporcionan las instrucciones SOUND y ENVELOPE. Éstas funcionan exactamente de la misma forma que lo hacen en el BASIC BBC.

Puede emplear COPY para editar líneas de la forma normal, y puede utilizar los códigos de control del BBC Micro, así como VDU, *FX y otras instrucciones *. Ambas versiones afirman soportar el segundo procesador (6502) y Econet. Ambas funcionaron bien con el Econet, pero no las probamos con el segundo procesador.

Escrita en BCPL y después compilada, la versión de Acornsoft se suministra en dos ROM de 16 K y viene acompañada de tres libros, una cinta y un disco de programas de muestra y ampliaciones del LOGO. Está colmada de características, teniendo más de 200 primitivas, incluyendo las de las ampliaciones en disco. La selección de las primitivas parace haber sido dictada por la motivación de incluir el mayor número posible de las mismas.



"Acornsoft LOGO", de Chris Jobson y John Richards; producido por Acornsoft Ltd, Betjeman House, 104 Hills Road, Cambridge CB2 1LQ, Gran Bretaña; basado en ROM

Analicemos las características extras más importantes. Un conjunto de primitivas permite la preparación de entornos modificados del LOGO, diseñándolos a la medida para el uso del estudiante. Hay dos características particularmente útiles: las primitivas se pueden redefinir y los procedimientos "enterrar" de modo que parezcan se primitivas; y se pueden preparar archivos que ejecuten un procedimiento de inicialización al cargarse.

"Gran parte de los trabajos iniciales en el campo de la inteligencia artificial utilizaron el LOGO, que guarda una estrecha relación con el LISP, el principal lenguaje para el diseño de sistemas expertos."

Para ayudar en la depuración de los programas, hay una amplia variedad de opciones de seguimiento: se puede rastrear cada línea, cada llamada a un procedimiento o cada primitiva, con la opción de hacer una pausa en cada punto. Esta nueva facilidad es excelente y, asimismo, sirve de ayuda para explicarle al estudiante la lógica de un programa. CATCH y THROW proporcionan una forma de saltar de un procedimiento a otro. Estas primitivas se utilizan fundamentalmente para detectar errores de entrada y, para ayudar en esta tarea, se facilitan otras numerosas primitivas.

"El LOGO posee dos tipos de objetos: 'palabras' y 'listas'. Asimismo, posee operaciones que permiten unir objetos, dividirlos en partes diferenciadas, o examinarlos."

Las ampliaciones que se suministran en disco ofrecen un número de facilidades adicionales. Hay un conjunto de primitivas para tratar con las "propiedades" (el LOGO LSCI Apple las posee), cuya idea deriva del LISP. La creación de múltiples tortugas (hasta 32), capaces cada una de ellas de tener una forma exclusiva y poder direccionarse de modo

individual, es factible mediante otro conjunto de primitivas. Hay ampliaciones que permiten impartir instrucciones de tortuga al BBC Buggy, la tortuga Jessop o la tortuga Valiant. Nosotros probamos la ampliación para el Buggy y nos pareció muy fácil de programar.

La versión Logotron se escribió en lenguaje máquina y viene en una única ROM de 16 K, con un manual de formación y de referencias en forma de carpeta de anillas. Fue desarrollada por SOLI (Systèmes d'Ordinateur LOGO International), quien en el pasado trabajó en estrecha relación con LCSI y fue también responsable del LOGO Spectrum. Esta versión es muy similar a cualquier otro LOGO LCSI.

"La característica más importante del LOGO es que puedes hacer que refleje tus intereses, tus necesidades y tu personalidad."

Con alrededor de 120 primitivas, SOLI ha optado de forma deliberada por restringir la cantidad de facilidades disponibles, con el objeto de darles cabida en una sola ROM. Ello supone una economía en el uso de ranuras para ROM y permite, asimismo, una mayor velocidad. La versión Logotron posee numerosas características extras que vale la pena destacar, una de las cuales permite escribir y leer programas en los archivos de texto soportados, al igual que el LOGO Atari. Se ha tenido gran cuidado al diseñar el editor de pantalla. Se han utilizado las teclas de función para controlar el editor y se han incorporado instrucciones FIND y REPLACE.

"Al LOGO se lo considera un 'lenguaje de programación para niños'. Pero también es un 'lenguaje de programación para científicos de ordenadores'."

Se proporcionan dos primitivas nuevas, OPPS y OPNS, que producen listas de nombres de procedimientos y nombres de variables, respectivamente. Éstas se pueden emplear como alternativas para las instrucciones más normales POTS y PONS. Una facilidad muy útil permite pasar variables a las instrucciones del OS BBC. Ésta permite que usted escriba procedimientos de nombre coherente para reemplazar los crípticos mensajes que exige normalmente el OS BBC.

La ausencia de dos primitivas resulta notable: TEXT y DEFINE. En el manual se ofrecen definiciones para ellas, pero dado que operan escribiendo en un archivo y volviéndolo a leer, obviamente son lentas. A los programadores estructurados les agradará saber que, tal como sucede en los LOGO Spectrum y Atari, se ha previsto la falta de espacio mediante la inclusión de una sentencia GO. La instrucción TRACE imprime el nombre de cada procedimiento, así como sus entradas, pero al abandonarlo no se produce ningún mensaje. Hubiera sido muy útil una opción TRACE completa.

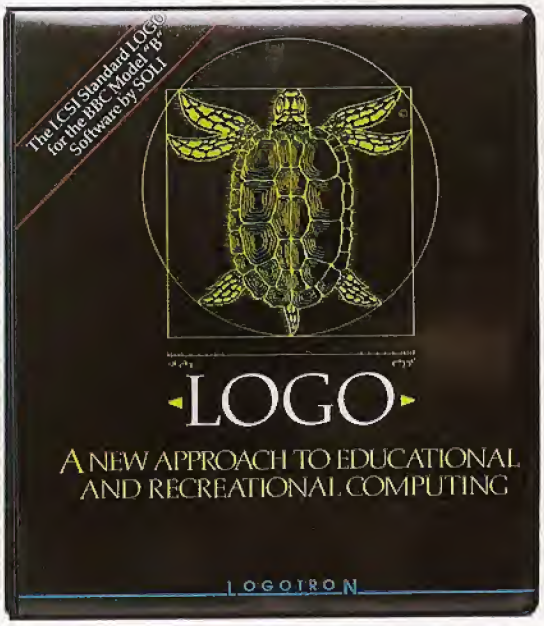
Ejecutamos varios programas de comparación (*benchmark*) con el fin de ofrecer algunas comparaciones entre las dos versiones desde el punto de vista de espacio de trabajo y velocidad. La versión LCSI posee un poco más de espacio de trabajo para definir variables y procedimientos, y alrededor del doble de espacio para la "pila", permitiendo, por consiguiente, más niveles de recursión antes de agotar la memoria. Ambas versiones implementan eficazmente la *recursión sin fin* para instrucciones y operaciones. Esto significa que los procedimientos de recursión sin fin continúan de forma indefinida.

En diversas pruebas de velocidad, la versión de Acornsoft demoró entre un 50 y un 1 000 % más que la versión Logotron, descubriéndose las diferencias más significativas en el proceso de listas y las asignaciones. (En la versión de Acornsoft, el recolector de basuras pareció especialmente lento.) Esta lentitud del proceso de listas de la versión Acornsoft podría resultar muy irritante; sin embargo, un lenguaje lógico similar al PROLOG, escrito en LOGO y que se suministra en el disco, se ejecutó con una velocidad aceptable.

Ambas versiones se entregan con manuales detallados que incluyen secciones de aprendizaje para principiantes y secciones de referencia. Éstas ofrecen explicaciones exhaustivas de geometría de tortuga, aunque en la versión de Acornsoft la distinción entre instrucción y operación no se expone con claridad. Cada uno de los manuales dedica apenas ocho páginas para describir el proceso de listas, pero la versión de Logotron compensa este hecho incluyendo los listados de algunos procedimientos gracias a los cuales a los estudiantes les resultará mucho más fácil aprender el proceso de listas. (Estos procedimientos son el resultado de los trabajos realizados en el Departamento de Inteligencia Artificial de la Universidad de Edimburgo.)

Acornsoft proporciona un buen número de extensos programas de demostración en disco (o cassette), incluyendo un programa tipo PROLOG y un ejemplo de análisis de "lenguaje natural".

Acornsoft ha prometido más ampliaciones para el futuro. Logotron confía en comercializar pronto un "Advanced LOGO" (LOGO avanzado) en disco, que le añadirá al lenguaje muchas primitivas. Más interesante aún es su promesa de una placa de sprites que configurará 30 sprites por hardware.



Ian McKinnell

"Logotron LOGO", de Gerard Dehan, Armen Valian, Marie Paule Deprund y Eric Millet; producido port Logotron Ltd, Ryman House, 59 Markham Street, London SW3, Gran Bretaña; basado en ROM

Intercambios básicos

Veamos ya qué ocurre entre el BASIC y el sistema operativo del BBC Micro

Esta serie de capítulos se abrió con una ideas sobre los sistemas operativos que hablaban del modo en que un OS nos previene contra futuros cambios en el hardware del ordenador. No puede, pues, sorprender que el BASIC utilice tan profusamente las llamadas OS. La instrucción en BASIC SOUND, por ejemplo, genera una llamada a OSWORD con A=7 para producir sonido; igualmente ENVELOPE emplea OSWORD CON A=8. Lo interesante es notar que podemos modificar el modo en que se comporta el BASIC, dentro de unos límites, cambiando el modo en que se comporta el sistema operativo, es decir, alterando los vectores y escribiendo nuestros propios fragmentos de código máquina que reemplacen las rutinas OS.

Asimismo, OSWRCH, OSASCII y OSNEWL se emplean en varias rutinas que acceden a la pantalla. Hay que observar que las rutinas en código máquina que producen visualizaciones de gráficos por medio de rutinas OS no siempre son mucho más rápidas que las instrucciones equivalentes en BASIC. Tenemos aquí una medida de la eficacia del empleo que hace el intérprete del BASIC de las llamadas OS a la hora de interpretar llamadas tales como MOVE, DRAW y PLOT.

Analicemos dos instrucciones BASIC empleadas para llamar desde el BASIC a programas en código máquina, ya sean rutinas del OS, ya sean programas propios. Son las instrucciones USR y CALL.

La llamada USR

El papel que desempeña la instrucción USR es el de ejecutar un fragmento de código máquina y devolver un valor al BASIC que nos dé detalles del estado de los registros A, X e Y después de haber ejecutado el código máquina. También se recaba información sobre el estado del flag de arrastre y del registro indicador de estado del procesador (PSR). Se llama así:

```
result%=USR(direcc%)
```

donde direcc% es la dirección de la rutina que hay que ejecutar. Antes de generar esta llamada se pueden poner valores en las variables A%, X% e Y%, que se pasarán, por medio de la rutina USR, a los registros A,X e Y, respectivamente. El bit menos significativo de la variable C% puede también utilizarse para afectar el valor del flag de arrastre; si C% es impar, éste se pondrá a cero.

El siguiente programa muestra el uso de USR en un breve ejemplo, donde llamamos a una de las llamadas OSBYTE que devuelve un valor en los registros X e Y. Esta rutina da el valor de la abscisa *x* del cursor del texto en el registro X, y el de la ordenada *y* de dicho cursor en el registro Y. El valor contenido en result% representa estos registros en forma codificada, de la que hablaremos luego.

```
10 A%=134
20 X%=0
30 Y%=0
40 LET result%=USR(&FFF4)
50 PRINT result%:REM imprime la representacion hexa
```

La impresión en hexadecimal del resultado nos facilita la decodificación de los valores dados por la llamada OSBYTE.

Función USR en acción

La función USR permite al usuario llamar desde el BASIC una sección del código máquina designada para devolver un valor al programa en BASIC. Dado que USR es una función (al contrario de CALL, que es una sentencia) y devuelve un valor, el resultado debe ser asignado dentro del programa en BASIC o a una variable:

```
result%=USR(&FFF4)
```

o como parte de una sentencia PRINT:

```
PRINT USR (&FFF4)
```

El resultado devuelto al BASIC por la función USR toma la forma de un número entero de cuatro bytes que refleja el contenido de los cuatro registros (P,Y,X y A) como sigue:

result%=

Reg. indic. de estado del proces.		Registro Y		Registro X		Registro A	

Byte más significativo — Byte menos significativo

Podemos obtener el valor deseado para un registro dado, empleando el operador AND que enmascara ciertas partes del resultado. Por ejemplo:

```
result%AND&000000FF
```

pondrá una máscara a cualquier valor excepto el del registro A

La instrucción CALL

El otro método para llamar programas en código máquina es la instrucción CALL. Ya la hemos empleado en varias ocasiones, en concreto, para llamar a OSWORD y otras rutinas del OS. En esos casos sólo llamábamos a la rutina especificando su dirección. Así,

```
CALL &FFF1
```

El BASIC del BBC permite también una versión ampliada de esta instrucción, que permite el acceso de un programa en código máquina a los valores rete-

nidos en las variables BASIC. Es obvio que el problema principal de tal organización es cómo decir al programa en código máquina dónde puede encontrar las variables en la memoria. Por suerte, el BASIC resuelve este problema gracias al intérprete; si hacemos una CALL con parámetros tal como

```
CALL direcc%,A%,C%
```

todos los detalles sobre los parámetros (en este caso, A% y C%) se almacenan en la página 6 de la memoria. Esta área de la memoria se conoce como *bloque de parámetros*, y el siguiente cuadro ilustra la organización de dicho bloque.

Posición	Descripción del contenido
&0600	Número de parámetros
&0601	Dirección del primer parámetro
&0602	
&0603	Tipo del primer parámetro
&0604	Dirección del segundo parámetro
&0605	
&0606	Tipo del segundo parámetro

Naturalmente, las entradas de la dirección y el tipo se hacen tantas veces como son necesarias. La dirección del parámetro es casi inmediata; se halla donde encontraremos el primer byte del parámetro en la memoria. La entrada del tipo indica qué tipo de variable es el parámetro. Este otro cuadro describe el abanico disponible de tipos de parámetros:

Tipo	Descripción	Ejemplo
0	Byte de ocho bits	CALL direcc,?&70
4	Variable entera	CALL direcc,F%
8	Variable real	CALL direcc,G
128	Serie en una direcc.	CALL direcc,$(&A00)
129	Variable en serie	CALL direcc,F$

La entrada final, tipo de parámetro 129, es ligeramente diferente: no es del todo exacto que la entrada de la dirección en el bloque de parámetros apunta a la posición de la serie en la memoria. De hecho apunta a un área de memoria que nos proporciona más detalles sobre la serie. Esta área de memoria se llama *bloque de información de la serie* y se organiza de esta manera:

Posición	Descripción de contenidos
Dir. contenida en bloque parám.	Dirección de la serie
	Bytes ubicados
	Longitud de la serie

Así, los primeros dos bytes del bloque de información apuntan al primer carácter en la serie. La entrada "bytes ubicados" mostrará la longitud máxima asignada a la serie antes de su longitud efectiva, que se encuentra en el cuarto byte.

El pase de parámetros

La sentencia CALL se emplea para ejecutar una sección de código máquina desde el BASIC, pero, a diferencia de la función USR, no devuelve un resultado. El usuario puede, no obstante, incluir una subrutina que pase los valores del programa en código máquina a las variables BASIC, permitiendo su lectura.

El siguiente programa no sólo da un ejemplo de pase de parámetros por medio de la sentencia CALL, sino que también pone un resultado en la variable entera A%. Además, ilustra el modo en que, al generarse errores en programas en código máquina, éstos puedan ser tratados desde el BASIC. Está escrito para BASIC II.

```
 10 DIM C400
 20 cero=&70
 30 FOR I%=0 TO 2 STEP 2
 40 P%=C
 50 [OPT I%
 60 .code LDA &600
 62 CMP #1
 64 BNE error
 70 LDA &603
 80 CMP #4
 90 BNE error2
100 LDA &601:STA &70
110 LDA &602:STA &71
120 LDY #0
130 LDA &02:STA(cero),Y:INY
140 LDA &03:STA(cero),Y
160 RTS
170 .error BRK
180 EQUB 254
190 EQUS "Numero incorrecto de
    parametros"
200 EQUB 00
210 .error2 BRK
220 EQUB 253
230 EQUS "Tipos equivocados"
240 EQUB 00
250 ]:NEXT
260 A%=0:CALL code,A%
270 PRINT "Fin de las variables
    BASIC",A%
```

El programa devuelve la dirección del fin de las variables BASIC en una variable entera pasada allí como un parámetro, demostrando así que CALL puede también devolver valores al BASIC. Las líneas 60 a 64 comprueban si sólo hay un parámetro, y si esto no es así se emite un oportuno mensaje de error. Las líneas de la 70 a la 90 comprueban entonces que el parámetro pasado a la rutina es un tipo entero, generándose otro mensaje de error si no es así. De la 100 a la 160 transfieren los datos de las direcciones 2 y 3 a los bytes menos significativos de la variable entera que se pasó como un parámetro. Las líneas de la 170 a la 240 sirven para generar los mensajes de error.

En la línea 260 se hace el CALL; pero antes se pone A% a cero porque el programa en código máquina no altera los bytes más significativos de la variable entera en cuestión.

Después de la llamada, la expresión HIMEM-A% dará el número de bytes que quedan cuando el programa en BASIC y las variables han ocupado su parte de memoria.

Conciencia visual

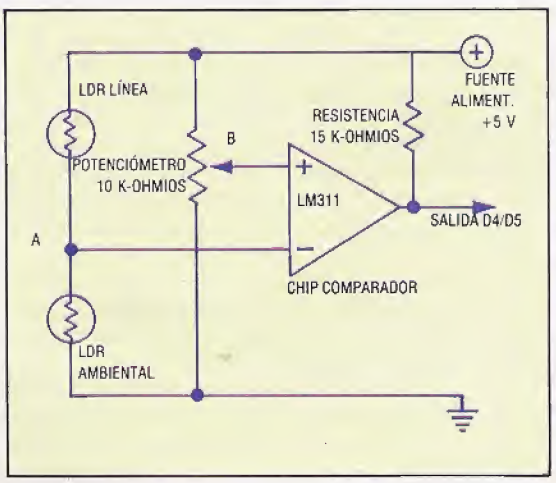
Llegados a este punto, dotaremos al robot con la "vista" necesaria para seguir una línea oscura

La unidad que le permite al robot seguir una línea recta está compuesta por dos circuitos idénticos, alimentando cada uno una línea de entrada diferente de la puerta para el usuario. Cada circuito se basa en un chip comparador LM311, que compara los voltajes de sus dos patillas de entrada. Si el voltaje de la patilla positiva es mayor que el de la entrada negativa, la salida saltará de cero voltios al voltaje de alimentación. En términos lógicos, la salida de este chip será cero si los dos voltajes de entrada son iguales, y uno si el voltaje de la patilla de entrada positiva es mayor que el de la patilla de entrada negativa.

Dos resistencias que dependen de la luz (LDR) en cada circuito forman un divisor de voltaje entre el voltaje de alimentación y tierra. Si la luz que cae sobre los dos LDR es la misma, entonces su resistencia también será igual y el voltaje en el punto A del diagrama de circuito será de unos 2,5 V (la mitad del voltaje de alimentación). El potenciómetro se puede regular para que dé el mismo voltaje en el punto B. Puesto que los dos voltajes de entrada son iguales, la salida del chip comparador será cero.

Si la luz que cae sobre la línea LDR disminuye (p. ej., cuando se halla sobre una línea oscura), su resistencia aumentará, haciendo que el voltaje del punto A sobrepase los 2,5 V. Esto, a su vez, hará que la salida del comparador salte al voltaje de alimentación, dando una salida lógica de uno e indicando que la línea LDR se halla sobre la línea oscura.

La presencia de dos circuitos, compuestos cada uno por un par de LDR y un chip comparador, permite que el ordenador detecte hacia qué lado de la línea se ha inclinado el robot. En cada par de LDR, uno mide la intensidad de la luz ambiental de modo que se pueda comparar con la intensidad de la luz medida por el otro LDR sobre la línea.



Bajando la vista
El circuito seguidor de líneas consta de dos circuitos idénticos, utilizando uno de ellos un par de resistencias que dependen de la luz. Se emplea un chip comparador para producir una señal digital si sobre los LDR están cayendo cantidades diferentes de luz. De esta forma, el robot puede detectar si se halla o no sobre una línea negra

Calibrado

Después de construir e instalar la placa, debemos calibrarla. En condiciones luminosas similares a aquellas en las que usted desea ejecutar el programa seguidor de línea, ejecute este programa de calibrado y regule los dos potenciómetros preestablecidos de la placa recién instalada.

Programas de prueba LDR

```
1000 REM **** PRUEBA LDR BBC ****
1010 RDD=&FE62:REGDAT=&FE60:?RDD=15:REM LINEAS 0-3
     SALIDA
1020 REPEAT
1030 contenido=?REGDAT
1040 IF(contenido AND 16)=0 THEN PRINT TAB(5)"IZQUIERDA";
1050 IF(contenido AND 32)=0 THEN PRINT TAB(15)"DERECHA";
1060 PRINT
1070 UNTIL FALSE
```

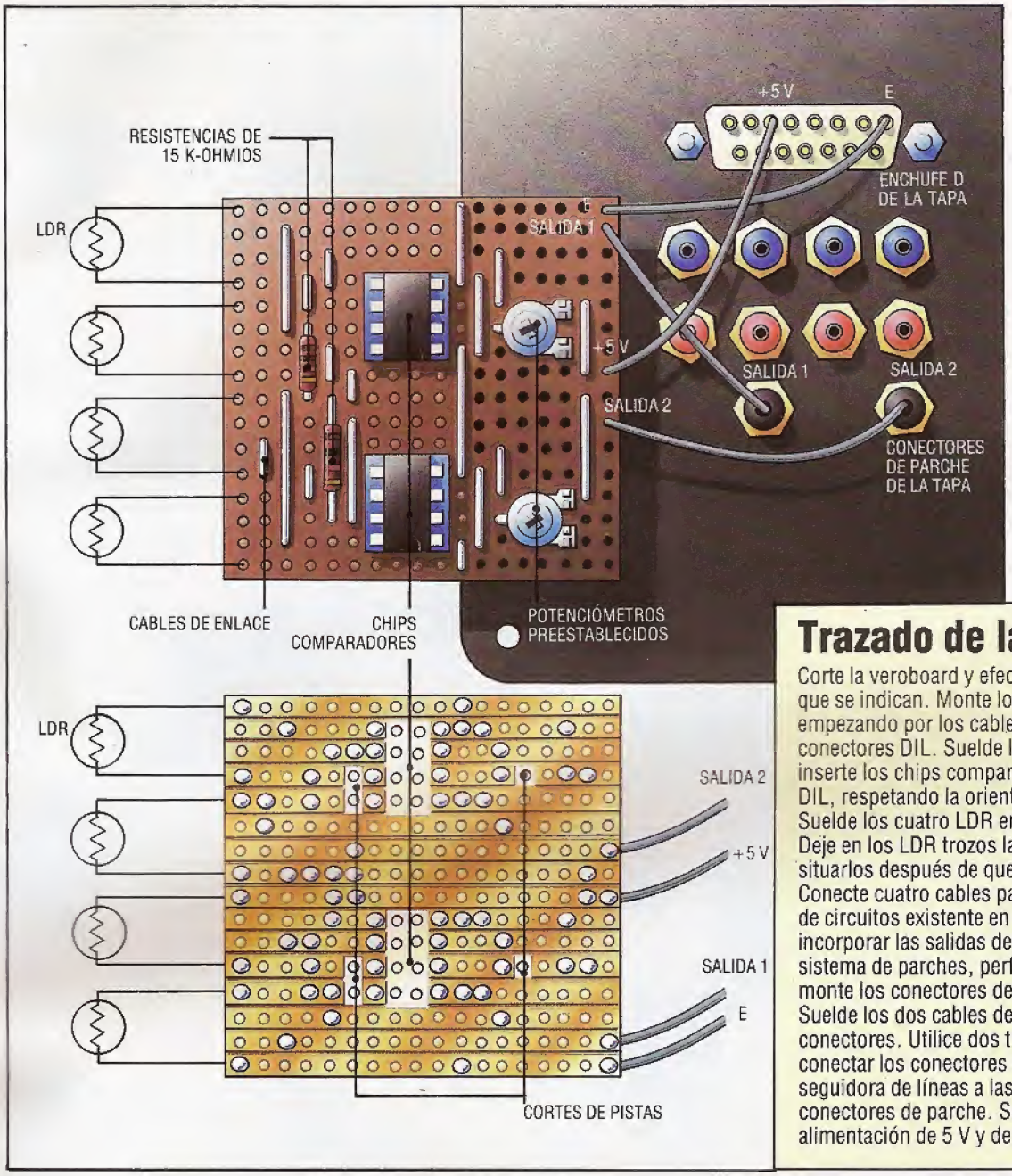
```
1000 REM **** PRUEBA LDR CBM ****
1010 RDD=56579:REGDAT=56577:POKE REGDAT,15
1020 CN=PEEK(REGDAT)
1030 IF(CN AND 16)=0 THEN PRINTTAB(5)"IZQUIERDA";
1040 IF(CN AND 32)=0 THEN PRINTTAB(15)"DERECHA";
1050 PRINT
1060 GOTO 1020
```

```
1000 REM **** PRUEBA LDR SPECTRUM ****
1010 CLEAR 32499:LET ST=32500:GO SUB 3000
1020 LET NM=16:GO SUB 2000:IF USR ST=0 THEN PRINT TAB
     5;"IZQUIERDA";
1030 LET NM=32:GO SUB 2000:IF USR ST=0 THEN PRINT TAB
     5;"DERECHA";
1040 PRINT
1050 GO TO 1020
1060 :
2000 REM **** REALIZAR AND ****
2010 POKE ST+1,IN 31
2020 POKE ST+3,NM:RETURN
2030 :
3000 REM **** CARGADOR CODIGO MAQUINA ****
3010 FOR I=ST TO ST+8
3020 READ A:POKE I,A
3030 NEXT I
3040 DATA 62,0,14,0,161,6,0,79,201
3050 RETURN
```

Marque en un trozo de papel blanco una línea negra de 2,5 cm de ancho. Tomando cada potenció-

Lista de componentes

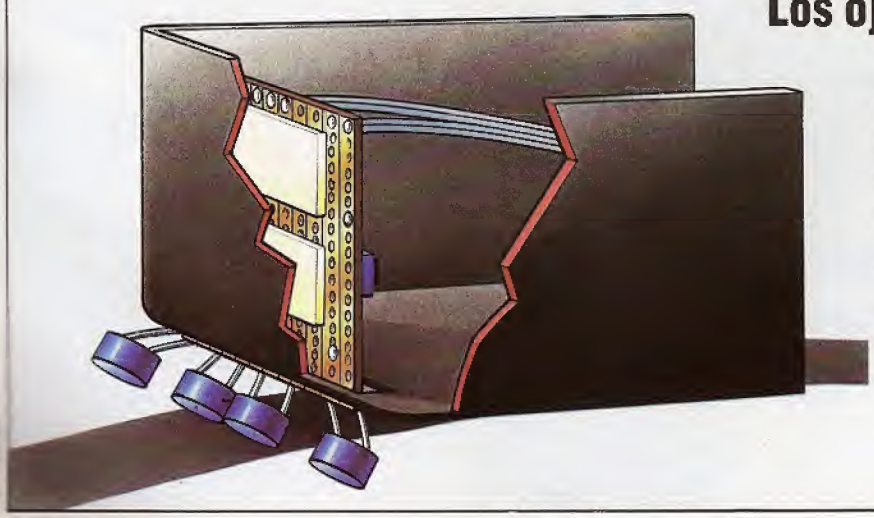
Cantidad	Artículo
2	Chip comparador LM311
2	Conector DIL de 8 patillas
2	Conector 2 mm
2	Resistencia 0,4 W 15 k-ohmios
2	Potenciómetro preestablecido horizontal 10 K
4	LDR ORP12
1	Veroboard 50 agujeros por 24 franjas
1	Rollo parches autoadhesivos
1 m	Cable plano 4 vías
1 m	Alambre estañado pelado 20 swg



Trazado de la placa

Corte la veroboard y efectúe los cortes de pistas que se indican. Monte los componentes pasivos, empezando por los cables de enlace y los conectores DIL. Suelde las dos resistencias e inserte los chips comparadores en los conectores DIL, respetando la orientación de las muescas. Suelde los cuatro LDR en el margen de la placa. Deje en los LDR trozos largos de cable para poder situarlos después de que la placa esté colocada. Conecte cuatro cables para enlazar con el sistema de circuitos existente en la tapa del robot. Para incorporar las salidas del chip comparador al sistema de parches, perfore dos nuevos agujeros y monte los conectores detrás del grupo de ocho. Suelde los dos cables de salida en estos conectores. Utilice dos trozos de parche para conectar los conectores de salida de la placa seguidora de líneas a las líneas 4 y 5 del grupo de conectores de parche. Suelde los cables de la alimentación de 5 V y de tierra al enchufe tipo D

Los ojos del robot



La placa del circuito seguidor de líneas se monta en la cara delantera interior de la carcasa del robot, de modo que los cuatro LDR sobresalgan a través de una ranura de 6×1 cm cortada en la base. Incline cuidadosamente las patas de alambre de los LDR para que los LDR estén en la formación que se indica. Los dos LDR centrales se deben empujar de modo que queden lo más cerca posible uno del otro, para que queden situados encima de la línea a seguir

Kevin Jones

metro de uno en uno, ajústelo con un pequeño destornillador de modo que el ordenador no dé ninguna salida cuando los dos LDR centrales se hallen sobre la línea, pero dé la salida correcta cuando el robot se desplace hacia alguno de los lados de la línea. Tras el ajuste compruebe la función desplazando lentamente el robot hacia los lados, a través de la línea, de izquierda a derecha. Cuando los cuatro LDR estén sobre el papel blanco, en la pantalla deberá visualizarse IZQUIERDA DERECHA. Cuando el robot se desplace sobre la línea, el mensaje deberá rezar IZQUIERDA, y cuando no se produzca ningún mensaje el robot estará colocado directamente sobre la línea. Cuando el robot se incline hacia la derecha, la pantalla deberá visualizar DERECHA y, por último, IZQUIERDA DERECHA cuando los cuatro LDR estén otra vez sobre el papel blanco. El programa da por sentado que el LDR de la derecha está parcheado en la línea 4 y el LDR de la izquierda en la línea 5.

Las condiciones de luz operativas deben permanecer constantes durante el calibrado y en las posteriores ocasiones en que se utilice el programa seguidor de línea. Es una buena idea iluminar con intensidad la superficie operativa empleando una luz brillante de flexo, de modo que en futuras ocasiones se puedan reproducir las mismas condiciones luminosas.

Programa seguidor de línea

El programa seguidor de línea utiliza rutinas para mover y hacer girar al robot. El programa da por sentado que éste comienza estando sobre la línea. Emplea un bucle recursivo para desplazar el robot 1 mm hacia adelante y comprobar si se ha apartado de la línea. Esto se detecta cuando el bit 4 o el bit 5 del registro de datos de la puerta para el usuario se pone *low*. Si, supongamos, el bit 4 se pone *low*, ello indica que el robot se ha apartado de la línea hacia la izquierda; el programa, en consecuencia, hace que el robot efectúe un giro de 5° hacia la derecha antes de continuar. Del mismo modo, si el que se pone *low* es el bit 5, es porque el robot se ha desviado hacia la derecha, produciéndose a continuación un giro de 5° hacia la izquierda. El programa termina cuando ambos bits se ponen *low*, como sería el caso cuando el robot llegara al final de la línea.

El ángulo de giro, 5°, se obtuvo a través de la experimentación. Quizá usted deba ajustar la cifra para adecuarla a las líneas que haya dibujado y a sus curvas. Tal vez desee, asimismo, ampliar este programa de modo que el robot encuentre primero la línea y luego la siga. El sistema de parches permite establecer combinaciones de sensores para las cuatro líneas de entrada disponibles en el registro de datos, de modo que usted puede desarrollar sus propias aplicaciones.

Progs. seguidores de línea

BBC Micro

```
 10 REM **** SEGUIDOR DE LINEA BBC ****
 20 :
 30 PROCinicializar
 40 PROCseguir_linea
 50 END
 60 :
 70 DEF PROCseguir_linea
 80 REPEAT
 90 PROCmover(adelante,1)
100 PROCprobar_ldr
110 UNTIL banderafinal=1
120 ENDPROC
130 :
140 DEF PROCprobar_ldr
150 valldr=?REGDAT AND 48
160 IF valldr=32 THEN PROCgirar(derecha,5)
170 IF valldr=16 THEN PROCgirar(izquierda,5)
180 IF valldr=0 THEN banderafinal=1
190 ENDPROC
200 :
210 DEF PROCinicializar
220 RDD=&FE62:REGDAT=&FE60:?RDD=15
230 ?REGDAT=1
240 adelante=4:atras=2:izquierda=6:derecha=0
250 coef_id=3.34446:coef_ia=379/90
260 banderafinal=0
270 ENDPROC
280 :
290 DEF PROCmover(dir,distancia)
300 ?REGDAT=(?REGDAT AND 1)OR dir
310 impulsos=coef_id*ABS(distancia)
320 FOR I=1 TO impulsos:PROCimpulso:NEXT I
330 ENDPROC
340 :
350 DEF PROCgirar(dir,angulo)
360 ?REGDAT=(?REGDAT AND 1)OR dir
370 impulsos=coef_ia*angulo
380 FOR I=1 TO impulsos:PROCimpulso:NEXT I
390 ENDPROC
400 :
410 DEF PROCimpulso
420 ?REGDAT=(?REGDAT OR 8)
430 ?REGDAT=(?REGDAT AND 247)
440 ENDPROC
```

Commodore 64

```
  10 REM **** SEGUIDOR DE LINEA CBM ****
  20 :
  30 GOSUB2000:REM INIC
  40 GOSUB1000:REM SEGUIR LINEA
  50 END
  60 :
1000 REM **** SEGUIR LINEA ****
1010 DR=FW:DS=1:GOSUB2500:REM MOVER
1020 GOSUB1500:REM PROBAR LOS LDR
1030 IF EF<>1 THEN 1000
1040 RETURN
1050 :
1500 REM **** PROBAR LOS LDR ****
1510 LV=PEEK(REGDAT)AND 48
1520 IF LV=32 THEN DR=RT:DS=5:GOSUB3000:REM GIRAR
1530 IF LV=16 THEN DR=LF:DS=5:GOSUB3000:REM GIRAR
1540 IF LV=0 THEN EF=1
1550 RETURN
1560 :
2000 REM **** INIC ****
2010 RDD=56579:REGDAT=56577:POKERDD,15
2020 POKEREGDAT,1
2030 FW=4:BW=2:LF=6:RT=0
2040 PD=3.34446:PA=379/90
2050 EF=0:REM BANDERAFINAL
2060 RETURN
2070 :
2500 REM **** MOVER (DR,DS) ****
2510 POKE REGDAT,(PEEK(REGDAT) AND 1)OR DR
2520 PL=PD*ABS(DS)
2530 FOR I=1 TO PL:GOSUB 3500:NEXT I
2550 RETURN
2560 :
3000 REM **** GIRAR (DR,DS) ****
3010 POKE REGDAT,(PEEK(REGDAT)AND 1)OR DR
3020 PL=PA*ABS(DS)
3030 FOR I=1 TO PL:GOSUB3500:NEXT I
3040 RETURN
3050 :
3500 REM **** IMPULSO ****
3510 POKE REGDAT,PEEK(REGDAT)OR 8
3520 POKE REGDAT,PEEK(REGDAT)AND 247
3530 RETURN
```

Sinclair Spectrum

```
  10 REM **** SEGUIDOR DE LINEA SPECTRUM ****
  12 REM SUPRIMIR LINEAS 2010,2020,2510,3010 DE LA VERSION CBM
  13 REM E INTRODUCIR ESTAS MODIFICACIONES
  15 CLEAR 32499:ST=32500:GOSUB4500
1510 LET NM=48:GO SUB 4000: LET LV=USR ST
3510 OUT 31,DR+9
3520 OUT 31,DR+1
4000 REM **** REALIZAR AND ****
4010 POKE ST+1,IN 31
4020 POKE ST+3,NM:RETURN
4500 REM **** CARGADOR DE CODIGO MAQUINA ****
4510 FOR I=ST TO ST+8
4520 READ A:POKE I,A
4530 NEXT I
4540 DATA 62,0,14,0,161,6,0,79,201
4550 RETURN
```

Viajar a otros mundos

Terry Kennett

La magia de aprender
Los juegos de aventuras y simulaciones de carácter educativo permiten que el niño explore, en el propio entorno de la clase, situaciones cuya observación en la vida real es inaccesible o potencialmente peligrosa. Lejos de ser una pérdida de tiempo, la experiencia sugiere que tales juegos pueden favorecer considerablemente el proceso del aprendizaje

Los juegos de simulación y de aventuras permiten al estudiante acceder a esferas que, de otra forma, serían inalcanzables para él

Los juegos son divertidos, y todos los niños y la mayoría de los maestros coinciden en que el aprendizaje también debería ser entretenido. La utilización de juegos en general y la representación de papeles en particular se han popularizado en los últimos años, paralelamente al auge que han experimentado las actividades teatrales en la educación y el desarrollo de la "teoría del juego" en psicología. Estos desarrollos le confieren aún más credibilidad al potencial del ordenador en la escuela.

El juego del tipo "marcianitos" y la más reciente variedad de los juegos de aventuras/simulación representan los dos géneros dominantes de los juegos por ordenador. A medida que se va desarrollando el software, las diferencias entre ambos van volviéndose cada vez más sutiles, pero la mayoría de los juegos todavía se pueden reconocer como inspiradores en uno u otro tipo de programas.

Aquellos pertenecientes a la variedad de conflictos espaciales tienen un impacto muy leve en el software educativo, a pesar de su popularidad. Al escribir su libro *Mind at play. The psychology of video games*, Elizabeth y Geoffrey Loftus interrogaron a numerosos psicólogos acerca de la efectividad educativa de esta clase de juegos: las respuestas que obtuvieron no fueron alentadoras. Como cabe esperar, el valor de los juegos de "marcianitos" como fuente de entretenimiento sobrepasa cualquier posible valor educativo, si bien la investigación demuestra que esta clase de juegos, debido a la coordinación y los reflejos que exige, puede tener efectos terapéuticos para quienes sufren de diversos tipos de enfermedades mentales y lesiones cerebrales.

Programas como *Spell invaders* (Los invasores de la ortografía) y *Maths invader* (El invasor de las matemáticas) han sido intentos de presentar temas tradicionales en formato de "invasores del espacio", teniendo el niño que "disparar" las respuestas correctas. Pero estos juegos han demostrado ser menos divertidos que los originales y su eficacia educativa es, cuanto menos, dudosa.

Cortesía de Simulation-Database Project, Glasgow College of

Ian McKinnell

Un viaje a través del tiempo
El Simulation-Database Project, desarrollado en el St. Andrew's College of Education de Glasgow, en Gran Bretaña, utiliza el ordenador como herramienta para complementar (y no sustituir) otras actividades del aprendizaje. El paquete se divide en tres partes: un "juego de aventuras", un conjunto de archivos de texto que proporcionan información adicional y una base de datos que contiene referencias a otras fuentes. La clase se divide en grupos, asumiendo cada grupo el papel de un personaje que (en este ejemplo) realiza un largo viaje a través de la Escocia del s. XIX. Las actividades adicionales que se derivan del uso de la base de datos, y que a la vez la acompañan, incluyen trabajo con mapas, escritura creativa, actividades teatrales y excursiones

Las simulaciones y las aventuras, por otra parte, demuestran un potencial considerable como alternativas educativas. Las simulaciones utilizan el ordenador para crear entornos de aprendizaje que de otra forma no estarían al alcance del niño, y los juegos de aventuras pueden reproducir los escenarios fantásticos de la ciencia-ficción y la mitología sin los gastos o los peligros inherentes a la realidad. Estos principios ya se han aplicado en los planes de entrenamiento industrial y en la actualidad muchos de ellos se están utilizando en las escuelas.

Ballooning (Aerostación), un programa de Hill McGibbon para el Spectrum y el Commodore 64, es un típico juego de simulación que le ofrece al niño la oportunidad de aprender jugando con algo a lo cual, de otra forma, jamás tendría acceso. En un panel de instrumentos situado en la parte inferior de la pantalla hay un indicador de combustible, un altímetro, un indicador de temperatura y un dial de velocidad de descenso/ascenso, que actúan de la forma adecuada cuando se enciende o se apaga el quemador, o cuando se abre y se cierra la válvula de gas. Cuando el globo aerostático desciende a una cierta altura se hace visible el terreno y, en el caso de que fuera necesario un reabastecimiento de combustible, hay varias estaciones de aterrizaje donde se podría hacer descender la nave.

Ventajas educativas

Además de ser económicas y seguras, las simulaciones ofrecen otras ventajas educativas. Representados en un ordenador, ya se han tenido en consideración los detalles de preparación de un experimento o ejercicio. Por consiguiente, los niños quedan libres para concentrarse en la situación, en lo que está sucediendo y por qué. El tiempo ya no es un obstáculo que impida completar en clase proyectos complicados y que consumen mucho tiempo. Experimentos científicos que pueden requerir días en el laboratorio se pueden observar en la pantalla en cuestión de segundos. Un modelo por ordenador simplifica los sistemas que por su complejidad resultan difíciles de comprender para los estudiantes, y los gráficos pueden realzar las funciones pertinentes con datos representativos de las etapas importantes del proceso. Dado que la simulación se puede repetir en circunstancias diversas, los estudiantes pueden experimentar de forma indefinida. La dirección en que avanza el experimento, y los factores variables implementados, al estar todos bajo el control del estudiante, permiten una mayor comprensión de los resultados y las conclusiones y, puesto que la simulación se presenta como un todo y no en partes, se obtiene una perspectiva más amplia.

Existen, no obstante, ciertas limitaciones inherentes a las simulaciones. Debido a que un ordenador puede almacenar sólo una cantidad predefinida de variables, existe el peligro de que una simulación presente una visión demasiado simplificada de su tema. Por consiguiente, la capacidad de describir situaciones de la vida real es al mismo tiempo su mayor cualidad y, tal vez, su mayor punto débil. En realidad, es lo imprevisto lo que nos sorprende y nos obliga a reevaluar nuestro conocimiento sobre lo que está sucediendo. Por este motivo, un programa como *Ballooning*, con su cantidad fija de variables, ofrecerá sus mayores ventajas cuando se lo utilice en un contexto más amplio, quizá como un único aspecto de la ciencia y la historia del vuelo. Lamentablemente, esta limitación con frecuencia se pasa por alto; una simulación no es un sustituto de la realidad. Su punto fuerte consiste en estimular a una mayor indagación de los temas relacionados, no en proporcionar respuestas ya elaboradas o como una simple ampliación de las técnicas de aprendizaje programado. Hay, por ejemplo, un

programa de ciencias escrito para el Apple, que simula un proyecto para la conexión de bombillas y pilas de diferentes maneras. Presumiblemente, se ha escrito para la escuela que pueda permitirse un ordenador caro pero que, por algún motivo misterioso, no tenga acceso ni a bombillas ni a pilas. El posible shock de una pila de 6 V apenas puede justificar la simulación por cuestiones de seguridad.

Creación de microcosmos

La mayoría de las personas, niños y adultos por igual, disfrutan recreándose en la fantasía. En los últimos años, libros como *El Hobbit, 2001: una odisea del espacio* y la serie *Conan,* de Robert E. Howard, han adquirido una notable popularidad. Juegos de representación de papeles como *Dungeons and dragons* (Calabozos y dragones), no sólo han obtenido un extraordinario éxito comercial, sino que también han coincidido con el masivo reconocimiento del valor de la representación de papeles en la terapia y la educación. Su naturaleza interactiva hace que el ordenador sea un medio ideal para los juegos de fantasía y actuación. Las capacidades de gráficos y sonido pueden definir de forma más cabal un mundo histórico o imaginario. Los especialistas en educación han sabido ver rápidamente este potencial del ordenador para crear microcosmos, considerándolos como un salto cualitativo en las prácticas de la enseñanza.

No parece, por ejemplo, haber forma mejor de que un niño aprenda historia que retrocediendo en el tiempo y viviendo una aventura. Pero el niño necesitará información para sobrevivir y, por tanto, el programa debe contener una base de datos que se pueda consultar durante el juego y utilizar como obra de referencia al margen del teclado. El Simulation-Database Project lo puso en marcha Allan Martin en el St. Andrew's College of Education de Glasgow, en Gran Bretaña, para "explorar la forma en que los ordenadores pueden contribuir significativamente al trabajo de la escuela primaria, mediante la creación de un nuevo tipo de objeto pedagógico basado en el ordenador: la base de datos de simulación". La estructura de la simulación está basada en un juego de aventuras: un niño entra en un mundo histórico en el cual, al viajar a través de diferentes zonas, ha de hacer frente a diversas situaciones e incidentes. Los archivos de texto de la base de datos están relacionados con cada una de las zonas y se puede acceder a ellos en cualquier momento. Los programas están diseñados para ser utilizados como parte de un proyecto mayor en el que la mayor parte del aprendizaje tendrá lugar fuera del ordenador, que es empleado sólo como uno entre muchos recursos.

Una segunda base de datos contiene referencias a la zona o el tema, y ésta puede ser consultada para obtener referencias a fuentes adicionales de información en libros, películas y otros medios de comunicación. Para ayudar a integrar el programa con otras actividades de la clase, el software incluye un libro de referencias, un mapa de la zona, cartas simuladas, imágenes y diverso material de referencia. Escritos en LOGO, los programas se pueden adaptar a las condiciones locales: es posible añadir zonas y datos, suprimirlos o alterarlos para adecuarlos al objetivo concreto de la escuela o clase en particular.

La primera base de datos simulada se basó en la Escocia de 1880. El paquete se utilizó en una clase dividida en seis grupos, asumiendo cada uno la identidad de un personaje ficticio pero inspirado en la realidad. Los personajes eran representativos de cada estrato de la sociedad, desde el aristócrata a la criada, y a cada grupo de alumnos se le dio un argumento que les exigía realizar un viaje a través de Escocia. Una vez que habían planificado su viaje con el mapa y se habían desplazado hasta la primera zona del ordenador, los personajes se tropezaron con diversos incidentes. Después de cada fase de movimiento, los niños, utilizando dos bases de datos y otras fuentes de referencias que les proporcionaba el maestro, investigaban la zona a la que habían llegado. Las actividades que generaban estas investigaciones incluían trabajo con mapas, redacción de diarios, redacción de informes de periódicos y otros trabajos descriptivos. Los niños producían obras, grababan "entrevistas" con los personajes, realizaban visitas a las áreas que cubría el juego e invitaban conferenciantes a la escuela. Favoreciéndose el debate durante todo el juego, así como el dibujo y la pintura de sus nuevos entornos, los estudiantes y el maestro disfrutaron con el desarrollo del ejercicio.

El proyecto de la base de datos de simulación constituye un ejemplo de cómo se puede utilizar un juego por ordenador para despertar en los niños el interés y estimular su trabajo en estudios sobre los temas relacionados.

El hombre y el juego

Muchos educadores y padres temen la posibilidad de que los estudiantes dediquen demasiado tiempo a "jugar" en lugar de "trabajar". Un reciente programa de televisión mostró a un grupo de niños explorando los gráficos de tortuga en el suelo de su clase. El comentarista advirtió a los telespectadores que no se debían dejar engañar por la naturaleza desestructurada del ejercicio al decir: "En realidad, aquí se está produciendo un verdadero aprendizaje."

No obstante, muchos padres y maestros están comenzando a creer que el "auténtico" aprendizaje sólo se produce a través del juego. Este punto de vista tan radical tiene una base clara, no sólo en teoría de la educación, sino también en filosofía e historia. Un libro de J. Huizinga (publicado originalmente en 1944, cuando era profesor de historia en la Universidad de Leyden, Alemania) ofrece un sólido argumento en el sentido de que el juego no es sólo un proceso de aprendizaje, sino que posee un valor sustancial a la cultura humana. La obra, titulada *Homo ludens* (derecha), resalta varios puntos sobre la importancia del juego:

- Es una función significativa: tiene un sentido
- Todo juego significa algo: dista mucho de ser una actividad inútil
- Es a la vez voluntario e instintivo. Los niños juegan instintivamente a aprender acerca del mundo que los rodea, pero siguen jugando porque les divierte
- La civilización contemporánea adolece de la formulación del juego: cuanto más estructurado sea éste, menos lúdico es

En la granja

"Farmfax" es un juego de programas que cubre cabalmente los principales aspectos de una granja moderna

Partiendo del hecho de que los ordenadores se están utilizando en todos los sectores de la sociedad y de la empresa, no sorprende en absoluto que se haya creado una selección de programas para granjeros. Lo que sí puede resultar sorprendente es que la mayoría de estos programas están destinados a una máquina que estuvo a punto de zozobrar, el Dragon, que ahora se fabrica en España y está adquiriendo gran popularidad entre los agricultores.

Uno de tales programas es la serie Farmfax, cuyo creador es el granjero Geoffrey Paterson. La aplicación práctica de cada paquete está orientada hacia una tarea específica de la granja, y la gente que escribió las especificaciones y las tradujo al lenguaje assembly sabía muy bien lo que hacía. Los programas están diseñados para ser fácilmente comprendidos por personas que no estén familiarizadas con los ordenadores y se basan en cartuchos de ROM que quedan listos para utilizar tras ser enchufados. Algunos de ellos poseen, asimismo, RAM extra en el cartucho para un almacenamiento de datos fácil y rápido, aunque es aconsejable que los usuarios conserven copias por razones de seguridad.

Un ejemplo de cómo el granjero puede sacar partido de la inclusión del micro en la administración de la granja es Robin Dunkerley, que administra una granja de 500 acres en las afueras de Oxford. Aproximadamente la mitad de sus tierras están reservadas para su rebaño de 150 Holsteins y Friesians, quedando la mayor parte del resto para el cultivo. Uno de los problemas que el programa Farmfax le ha ayudado a resolver es la determinación del "Índice de Calving" (el número de días que pasa una vaca sin parir y continuando, por consiguiente, su producción de leche). "Se ha comprobado que si una vaca no pare al menos una vez al año —explica—, a uno le cuesta unas 2 libras esterlinas por cada día de los 365 que ha estado sin parir. En el pasado, se nos han pasado 390 días o más; el promedio ha sido de 375. El ordenador nos ayuda a seguir el rastro de cualquier vaca que no haya sido entregada al toro y hasta ahora no nos ha fallado ni uno. El año pasado nuestro índice de Calving fue sólo de 366, y tengo mis dudas de si llegaremos alguna vez a conseguir uno mejor. Hemos alcanzado casi un índice óptimo."

Como indicador del ámbito de un programa, el *Individual cow records* puede retener todos los registros de hasta 240 cabezas de ganado dentro del cartucho; una versión más económica los almacena en cinta. Puesto que todos los datos se cargan en la memoria y se manipulan antes de volver a guardarse (SAVE), ambas versiones trabajan de forma idéntica. Sin embargo, aunque muchos considerarán la cinta como un medio más lento para el almacenamiento de datos, a los granjeros el tiempo les resulta bien invertido, ya que con anterioridad debían realizar tediosos cálculos para adoptar decisiones tales como cuándo vender una vaca cuya producción está disminuyendo y comparar lo que está costando en alimentos con lo que está produciendo en venta de leche. Y, lo que es más significativo, encuentran que es más eficaz que alquilar por horas un ordenador central.

"Solíamos pagar unas libras mensuales a un centro de cálculo para conservar nuestros registros en su ordenador", recuerda Dunkerley. "A fin de mes les enviábamos la información, y las salidas impresas nos llegaban unas tres semanas después, cuando ya era demasiado tarde para actuar en función de la información. Asimismo, gran parte del material de

la información que nos proporcionaban era valioso para quien tuviera una rebaño de mucho pedigree, pero era totalmente irrelevante para un rebaño lechero comercial como el mío, que se prepara tan sólo para producir leche." Durante mucho tiempo pensó que sería totalmente lógico adquirir su propia máquina; pero, sin embargo, Dunkerley no estaba preparado para pagar 2 500 libras. Pero el Dragon, la impresora y los programas Farmfax, cuyo precio completo oscilaba entre 600 y 700 libras, sí estaba al alcance de su presupuesto. Su gasto se amortizó rápidamente y ahora tiene acceso a la información que necesita cuando lo requiere, en vez de tener que pagar por información que con frecuencia le llegaba tarde y era irrelevante.

Ahora, al final de cada mes, Dunkerley pasa alrededor de una hora organizando y calibrando su información. Primero ejecuta el *Dairy management program* (programa para administración de lecherías), que le da resultados financieros, el margen de ingresos por encima de los gastos de alimentación, sus ingresos brutos, cuántos kilos de forraje está utilizando para generar cada litro de leche, la producción total de leche y otros datos importantes. Después ejecuta el *Dairy ration* para asegurarse de que está produciendo la cantidad debida de leche. Le resulta muy útil porque lleva a cabo un análisis de alimentación completo. Entrando la cantidad de megajulios de energía y cuánta materia seca y proteína cruda digerible hay en cada kilo de alimento, y entrando el precio y si se ha producido o no en la granja, Dunkerley puede determinar el costo total y el costo de cada vaca por día.

Entrando la producción diaria de leche de cada vaca, el programa *Individual cow records* proporcionará un registro de absoluta precisión al cabo de cada mes. "Si cambio algunos de los factores —explica Dunkerley—, el ordenador me vuelve a calcular todo automáticamente. Esto es lo bonito de un ordenador: gracias a él, el tiempo que lleva esta clase de trabajo se reduce a menos de un tercio. Y también me informa de cosas que a mí siempre me había interesado saber, por ejemplo, cómo marcha mi giro en descubierto bancario. Yo, como agricultor arrendatario, soy muy vulnerable al fatídico día, que se produce dos veces al año, en que vence mi alquiler. También puedo hacer un presupuesto anticipado exacto, lo que satisface en gran medida al gerente de mi banco."

Cada uno de los módulos del programa Farmfax opera sobre una base de gran amabilidad hacia el usuario, e incluye un manual, pequeño y simple, cuyo estudio sólo requiere unos minutos, tratándose, por lo general, de una hoja de papel plegada que cabe en el bolsillo. Todos ellos empiezan con un sencillo menú. Por ejemplo, *Cullcow*, que ayuda al granjero en aquellas decisiones relativas a considerar las vacas cuya producción de leche cae por debajo de un cierto nivel, primero le solicita al granjero que escoja entre:

PAGE 1 — DETALLES DE PRECIO
PAGE 2 — MEJOR COMPRA/VENTA

En la página 1, el granjero debe entrar primero el mes y el año a considerar, luego el tipo de interés bancario aplicable, el precio de la leche, la producción de leche en litros por día de cada vaca, y el costo mensual. Apenas se entran estos cuatro elementos, las mismas cifras se transfieren a todas las otras entradas para ahorrar tiempo de digitación, si bien se las puede alterar de forma manual si se piensa, por ejemplo, que las tasas de interés pueden variar en los meses siguientes.

En la página 2 se le pregunta al granjero si está comprando o vendiendo vacas, la cuantía del lote, el precio esperado y el mes de venta. El ordenador compara entonces todas las cifras y coloca una estrella junto a la vaca que representa la mejor oferta. Pulsando la tecla BREAK en cualquier momento el usuario regresará al índice, para volver a entrar cualquier dato si así fuera necesario.

Los otros paquetes operan sobre una base similar, si bien, como es obvio, en los programas más complejos el menú de apertura será más complejo. Por ejemplo, la página de apertura del *Dairy ration formulation* ofrece:

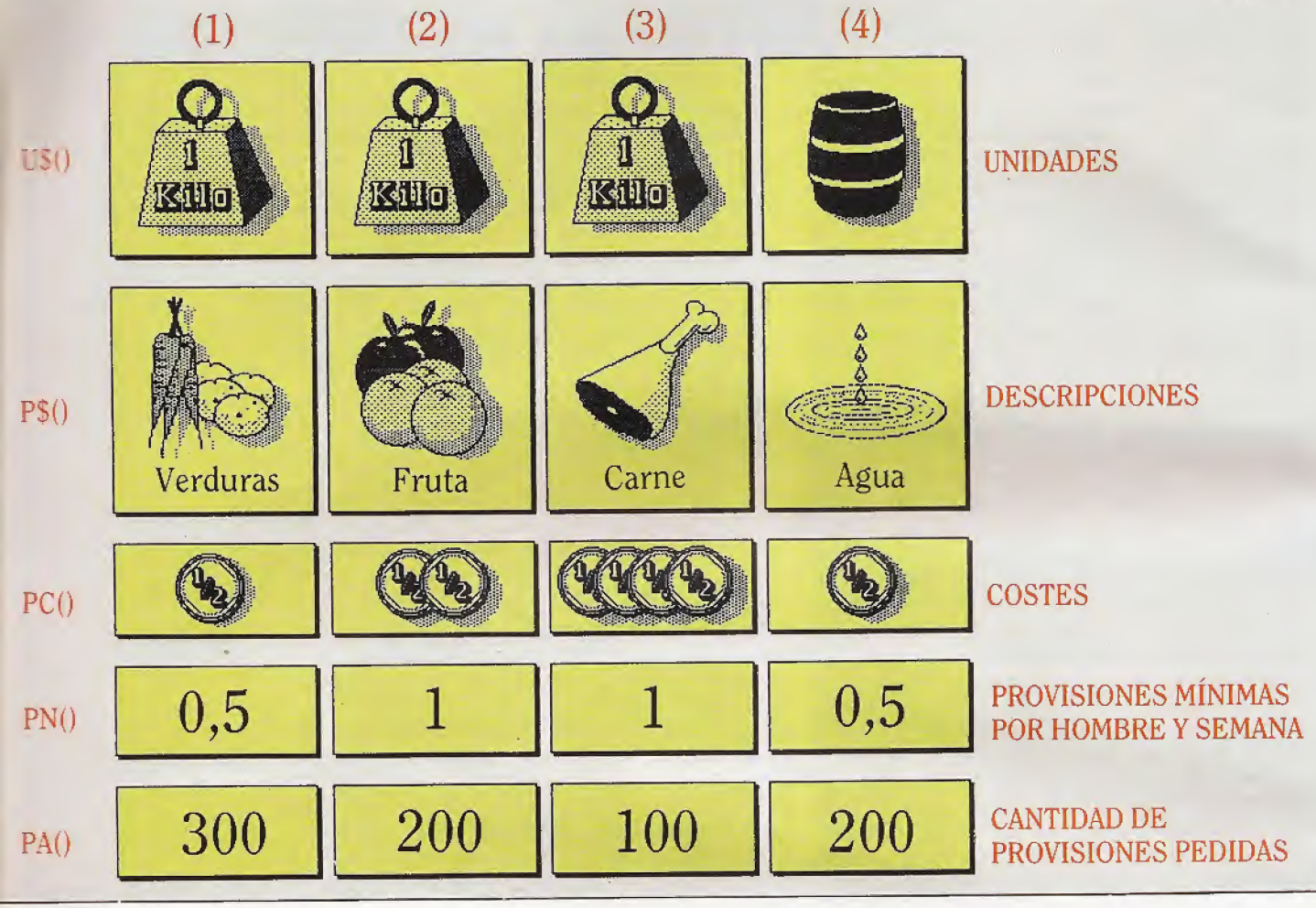
PAGE 1 — TABLA DE ALIMENTOS
2 — MINERALES
3 — COMPARABILIDAD
4 — DESCRIPCION DE GRUPOS
5 — FORMULACION
6 — CANTIDAD DE PIENSOS
7 — ARCHIVO/RECUPERACION

Si se emplea un cartucho sin memoria, el programa se activa automáticamente en la opción 7, permitiendo recuperar desde cinta datos anteriores; de lo contrario, se visualiza el menú. Cada uno de los alimentos entrados junto con sus precios en la página 1 se repite de forma automática en la página 2, en la cual el usuario puede entrar el contenido de minerales de cada uno, mientras que en la página 3 se analizan los datos, produciendo una evaluación de cuánta energía, cuánta materia seca y proteína suministra cada una por cada libra esterlina. Si el granjero entra en la página 4 sus objetivos de producción en términos de peso de la vaca, días de estar preñada, litros de leche producidos y contenido de grasa de la leche, entonces se puede, con la ayuda del programa para formular su proporción alimenticia, igualar al instante los costos totales de alimentación para un número dado de vacas durante una cantidad dada de días.

Evidentemente, ésta es la clase de programas escritos por especialistas y para especialistas. Los granjeros como Robin Dunkerley valoran la asistencia que este software ofrece en un mundo agrícola cada vez más industrializado y agobiado por la burocracia administrativa. "Con los cupos de la CEE y las presiones intensificándose cada vez más —concluye—, nos vemos obligados a llevar la granja de la forma más controlada posible, y el ordenador nos ayuda a hacerlo. No estoy seguro de si efectivamente nos ayuda a ganar más dinero, pero, por cierto, sí nos ahorra muchísimo."

Farmfax Suite: Para el Dragon
Distribuido por: Farmfax Computer Systems, PO Box No. 2, Stockbridge, Hampshire SO20 6LE, Gran Bretaña
Autor: Geoffrey Paterson
Formato: Cartucho

Ian McKinnell



Necesidades alimenticias
El jugador puede seleccionar cuatro tipos de provisiones: verduras, fruta, carne y agua. Las descripciones de estos cuatro tipos están almacenadas en la matriz P$() y las unidades correspondientes en la matriz US(). Para tomar las decisiones relativas a la cantidad de provisiones a adquirir, el jugador debe disponer de dos clases de información: el coste de cada provisión y las necesidades mínimas por semana por cada miembro de la tripulación. Estos datos están retenidos en las matrices PC() y PN(). Estas cuatro matrices se establecen al principio del programa. Hay otra matriz que se va llenando a medida que se encargan provisiones. PA() retiene la cantidad de cada provisión

Líneas de suministro

Nos corresponde desarrollar el segundo módulo del programa para el juego mercantil "Nuevo Mundo"

Al pulsar cualquier tecla aparece en pantalla la segunda etapa, en la que deben adquirirse las provisiones para la travesía. El jugador sabe que el viaje durará alrededor de ocho semanas, de modo que habrá que llevar una adecuada cantidad de alimentos y de agua.

El programa calcula las necesidades totales para ocho semanas y comprueba que se hayan adquirido provisiones suficientes. De no ser así, avisa al jugador, diciéndole que alguien podría padecer hambre o sed, y le ofrece la oportunidad de efectuar otro pedido. Si el jugador no dispone de oro suficiente para abonar el pedido, entonces el programa le indica que vuelva a probar (éste visualiza un balance de dinero durante la etapa de aprovisionamiento).

Al igual que en la etapa de contratación, necesitamos preparar algunas matrices:

```
19 REM **** MATRICES DE APROVISIONAMIENTO ****
20 DIMUS(4):US(1)="KILO":US(2)="KILO":US(3)="KILO":US(4)="BARRIL"
21 DIMP$(4):P$(1)="VEG":P$(2)="FRUTA":P$(3)="CARNE":P$(4)="AGUA"
22 DIMPA(4)
23 DIMPC(4):PC(1)=.5:PC(2)=1:PC(3)=2:PC(4)=.5
24 DIMPN(4):PC(1)=2:PN(2)=1:PN(3)=1:PN(4)=.5
```

Las provisiones se suministran en unidades diferentes: las verduras, la fruta y la carne se miden por kilos y el agua por barriles. La línea 20 prepara una matriz, U$, con cuatro elementos para retener estas cuatro unidades. Grabar las unidades en una matriz en vez de tener que imprimirlas continuamente representa un ahorro de memoria. Si desea ampliar la lista para incluir sacos de harina, por ejemplo, esta matriz se puede ampliar fácilmente.

Las descripciones de los cuatro tipos de provisiones están representadas en la línea 21 por los cuatro elementos de la matriz P$(). El programa prepara la matriz PA() para almacenar la cantidad adquirida de cada provisión. Un kilo de verduras cuesta media pieza de oro, un kilo de fruta una pieza de oro, un kilo de carne dos piezas de oro y un barril de agua media pieza de oro. Esta información se

almacena en la matriz de la línea 23. El orden de las provisiones no se enuncia en esta línea, sino que se establece por defecto. La matriz se puede ampliar fácilmente para incluir provisiones extras.

Para mantener un buen estado de salud, cada miembro de la tripulación necesita cierta cantidad de cada provisión. Estas necesidades se establecen en una matriz, PN(), en la línea 24. Por ejemplo, un miembro de la tripulación necesita dos kilos de verduras por semana, de modo que el primer elemento, PN(1), se establece en 2.

La línea 500 del primer módulo del programa envía el programa a la subrutina de la línea 1000 para contratar la tripulación. Tras la contratación, retorna a la línea 500. Para llamar a la rutina de aprovisionamiento hemos de insertar una llamada GOSUB después de la línea 500; de este modo:

```
550 GOSUB 2000: REM APROVISIONAMIENTO
```

Se visualiza una pantalla que explica el aprovisionamiento. La variable CN, establecida durante la etapa de contratación, representa el número de tripulantes contratados. Se utiliza en la línea 2040 para recordar cuántas bocas se han de alimentar.

La programación del aprovisionamiento se halla dentro de un bucle y se emplea cuatro veces, una vez para cada tipo de provisión. El valor del subíndice T se incrementa cada vez en uno. Éste se prepara en la línea 2050. La pantalla imprime la cantidad de cada tipo de alimento que necesita cada tripulante por semana. Estas necesidades se determinan usando el elemento correspondiente de PN().

Como hemos visto, los alimentos se miden por kilos y el agua por barriles. Las unidades requeridas se determinan mediante el elemento correspondiente de la matriz U$(), definida en la línea 20. Si se requiriera más de un kilo, la línea 2075 le añade a kilo una S; de ir en singular, la línea 2080 reemplaza la S por un espacio. La línea 2085 imprime DE y el nombre de la provisión retenida en P$(), seguido de POR CADA SEMANA QUE DURE EL VIAJE.

En las líneas 2100 y 2110 se le pregunta al jugador cuántas unidades de cada provisión desea comprar. Las demandas se digitan y se representan mediante PS() en la línea 2130. La línea 2140 coloca la cantidad en el primer elemento de la matriz de cantidades. Si la cantidad pedida es cero, el programa imprime SI NO COMPRAS NADA DE seguida de la matriz de provisiones P$(T) en la línea 2160. El programa calcula entonces si la cantidad pedida y entrada en la matriz PA() es mayor que la requerida para un viaje de ocho semanas, utilizando la fórmula CN*8*PN(T). CN es el número de tripulantes, y PN(T) corresponde a las necesidades de una provisión dada, determinada por T. Esto se produce en la línea 2170. Si las provisiones son insuficientes se imprime un mensaje de advertencia. El programa comprueba si está realizando el bucle por cuarta vez. De ser así, imprime SED en vez de HAMBRE.

Si se le ha advertido al jugador que las provisiones son insuficientes, el programa (en la línea 2230) le dice que: QUIERES PROBAR OTRA VEZ (S/N)? Si el jugador, a pesar de disponer de pocos fondos, acepta la entrada, el programa pasa a la línea 2400. Si desea hacer un pedido diferente, el valor pertinente de la matriz PA() se establece en cero. En este caso el programa ha de retroceder hasta el mismo tipo de provisión. Ello lo hace restándole 1 al contador del bucle mediante T=T−1, en la línea 2250.

Si el jugador está satisfecho con el pedido realizado, entonces se visualizan la cantidad de dinero que le queda y una lista de las provisiones pedidas hasta el momento. Puede ser que el jugador haya comprado provisiones suficientes para el viaje, pero que no le quede oro para abonar la factura. La línea 2260 comprueba si es éste el caso. De ser así, las líneas 2270-2290 indican que no hay dinero suficiente para comprar la cantidad de provisiones y se le solicita al jugador que vuelva a probar. Si la compra fue válida, su costo se resta del oro disponible, retenido en MO.

En la línea 2410, dentro de otro bucle controlado por TT, se imprime una lista de las provisiones adquiridas hasta el momento, utilizando las variables U$(), PA() y P$(). La cantidad de oro que aún queda se visualiza en la línea 2480. El bucle termina en la línea 2490. El jugador pasa a la siguiente etapa respondiendo al mensaje PULSE CUALQUIER TECLA PARA CONTINUAR. En este punto, el control retorna al programa principal.

Módulo dos

```
 000 REM **** APROVISIONAMIENTO ****
2005 PRINTCHR$(147):REM LIMPIAR PANTALLA
2010 S$="       ETAPA 2 - APROVISIONAMIENTO*"
2015 GOSUB 9100
2020 S$="       -------       ----------*"
2025 GOSUB 9100
2030 GOSUB 9200:PRINT
2040 PRINT"HAS CONTRATADO UNA TRIPULACION COMPUESTA POR
     ";CN;"MIEMBROS."
2045 GOSUB 9200:GOSUB 9200
2050 FOR T=1 TO 4
2055 PRINT
2060 PRINT"CADA MIEMBRO DE LA TRIPULACION NECESITARA "
2070 PRINT"AL MENOS ";PN(T);" ";U$(T);
2075 IFPN(T)=1THENPRINT" ";:GOTO2085
2080 PRINT"S";
2085 PRINT"DE";P$(T)
2086 PRINT"A";PC(T);"PIEZAS DE ORO POR ";U$(T)
2090 PRINT"POR CADA SEMANA QUE DURE EL VIAJE."
2095 GOSUB 9200:PRINT:GOSUB 9200
2100 PRINT"CUANTOS ";U$(T);"S DE ";P$(T)
2110 S$="DESEAS COMPRAR*":GOSUB 9100
2120 PRINT
2130 INPUTP$
2140 PA(T)=VAL(P$):GOSUB 9200
2150 IFPA(T)>((CN*8*PN(T))-1)THEN2260
2160 IFPA(T)=0THENPRINT"SI NO COMPRAS NADA DE":GOTO2180
2170 PRINT"SI SOLO COMPRAS ";PA(T);U$(T);
2175 IFPA(T)=1THENPRINT" DE":GOTO2180
2176 PRINT"S DE"
2180 PRINTP$(T);", ":GOSUB 9200
2190 PRINT"ALGUIEN PODRIA PASAR ";
2200 S$="HAMBRE"
2210 IF T=4 THEN S$=SED"
2220 PRINT S$;"!":GOSUB 9200
2230 S$="QUIERES VOLVER A PROBAR (S/N)":GOSUB 9100
2240 INPUTPS:PS=LEFT$(P$,1)
2242 IF P$ <> "S" AND P$ <> "N" THEN 2230
2245 IF PS="N" THEN 2400
2250 PA(T)=0:T=T-1:GOTO2410
2260 IFPA(T)*PC(T)> MOTHEN2270
2265 GOTO 2400
2270 S$="NO TIENES DINERO SUFICIENTE PARA":GOSUB 9100
2280 PRINTPA(T)
2290 PRINTU$(T);"S DE";P$(T):GOSUB 9200
2300 S$="POR FAVOR VUELVE A PROBAR*":GOSUB
     9100:PA(T)=0:T=T-1:GOTO 2410
2400 MO=MO-(PA(T)*PC(T))
2410 PRINT:S$="PROVISIONES HASTA AHORA:*":GOSUB 9100
2412 GOSUB 9200
2415 FOR TT=1 TO 4
2420 PRINT PA(TT);U$(TT)
2430 IF PA(TT)=1THEN PRINT" DE ";:GOTO2440
2435 PRINT"S DE ";
2440 PRINTP$(TT)
2450 GOSUB 9200
2460 NEXT
2480 PRINT"DINERO QUE TIENES AUN = ";MO;"PIEZAS DE ORO"
2485 GOSUB 9200:GOSUB 9200
2490 NEXTT
2500 GOSUB 9200:PRINT:S$="FIN DEL APROVISIONAMIENTO*":GOSUB 9100:
     GOSUB 9200
2510 PRINT: S$=K$:GOSUB 9100:PRINT: GOSUB9200
2520 GETP$:IFP$=""THEN2520
2999 RETURN
```

Líneas de abastecimiento
Esta subrutina se ocupa del aprovisionamiento para el viaje y debe ser añadida al primer módulo, que hemos ofrecido anteriormente. El programa pasa cuatro veces por un bucle, permitiendo que el jugador encargue una de las cuatro provisiones en cada pasada

Complementos al BASIC

Spectrum:
Introduzca las siguientes modificaciones:
2005 CLS
2240 INPUT P$:LET P$=P$(1 TO 1)

BBC Micro:
Introduzca las siguientes modificaciones:
2005 CLS

Alunizaje

Esta versión es para el Commodore 64. Un consejo: grabe el juego en cassette para evitar molestias

Después de un largo viaje sin gravedad, no resulta nada fácil alunizar suavemente con una nave espacial; pero gracias a su ordenador, está en condiciones de efectuar un entrenamiento sin riesgos. Debe colocar su nave sobre la zona destinada al efecto. Puede dirigirse hacia la derecha y hacia la izquierda con ayuda de las teclas de control del cursor, o bien frenar el descenso mediante la barra espaciadora. Para conseguir el alunizaje, las velocidades verticales y horizontales deben ser iguales o inferiores a 1.

```
  5 REM *************
 10 REM * ALUNIZAJE *
 15 REM *************
 20 GOSUB 1000
100 GET X$
105 IF FU=0 THEN 150
110 IF X$<>"" THEN FU=FU-1
120 IF X$="[1SPC]" THEN VV=VV-1
130 IF X$=G$ THEN VH=VH-1
140 IF X$=D$ THEN VH=VH+1
150 VV=VV+0.1
160 V=V+VV
180 H=H+VH
190 IF H>255 THEN H=0:MS=M1
200 IF H>90 AND MS=M1 THEN 4540
210 IF H<0 AND MS=M1 THEN H=255:MS=M0
220 IF H<0 AND MS=M0 THEN 4540
240 PRINT HO$;
250 PRINT "CARBURANTE[1SPC]:";STR$(FU); "[4SPC]"
255 IF V<0 THEN 300
260 POKE D+16,MS
265 POKE D,H
270 POKE D+1, INT(V)
275 IF V>V1-3 THEN 3000
300 PRINT "VELOCIDAD[1SPC]VERT.[1SPC]:";STR$
    (INT(10*VV)/10);"[4SPC]"
310 PRINT "VELOCIDAD[1SPC]HORIZ.:";STR$(VH);
    "[4SPC]"
320 GOTO 100
1000 D=53248
1005 RESTORE
1010 FOR I=0 TO 190
1020 READ A
1030 POKE 832+I,A
1035 NEXT I
1040 M1=5
1050 M0=0
1060 POKE D+39,7
1080 POKE D+40,7
1090 POKE D+41,0
1100 POKE 2040,13
1110 POKE 2041,14
1120 POKE 2042,15
1130 V1=230
1140 V0=0
1150 FU=90
1160 G$=CHR$(17)
1170 D$=CHR$(29)
1180 HO$=CHR$(19)
1190 M=54272
2000 PRINT CHR$(147);
2010 POKE D+21,0
2020 H=INT(RND(TI)*191)+65
2030 AH=INT(RND(TI)*191)+65
2050 FU=FU+10
2060 MS=M0
2070 VH=0
2080 VV=0
2090 V=0
2100 POKE D+2,AH
2110 POKE D+3,V1
2120 POKE D,H
2130 POKE D+1,V
2140 POKE D+21,3
2150 RETURN
3000 IF ABS(H-AH)>4 THEN 4000
3005 IF VV>1 THEN 4000
3010 IF ABS(VH)>1 THEN 4000
3015 FOR I=1 TO 4000
3020 NEXT I
3030 SC=SC+1
3040 GOSUB 2000
3050 GOTO 100
4000 POKE D+5,V+5
4010 POKE D+4,H
4020 POKE D+21,6
4500 FOR I=1 TO 5
4510 PRINT
4520 NEXT I
4530 PRINT TAB(6)"LA[1SPC]NAVE[1SPC]SE[1SPC]
    HA[1SPC]ESTRELLADO"
4540 IF SC>RE THEN RE=SC
4545 FOR I=1 TO 2000
4550 NEXT I
4555 FOR I=1 TO 3
4560 PRINT
4570 NEXT I
4580 PRINT TAB(13)"PUNTOS[1SPC]:";SC
4590 SC=0
5000 FOR I=1 TO 3
5010 PRINT
5020 NEXT I
5030 PRINT TAB(10)"PUNTUACION[1SPC]MAXIMA
    [1SPC]:"RE
5040 IF RE<SC THEN RE=SC
5050 SC=0
5060 FOR I=1 TO 3
5070 PRINT
5080 GET X$
5090 NEXT I
5100 PRINT TAB(13)"OTRA[1SPC]?"
5110 GET X$
5120 IF X$="" THEN 5110
5130 IF X$()"N" THEN POKE D+21,0:GOTO 20
5140 END
10000 DATA 0,255,0,1,255,128,3,255,192
10010 DATA 7,255,224,12,195,48,12,195,48
10020 DATA 15,255,240,12,102,48,12,102,48
10030 DATA 15,255,240,7,255,224,3,129,192
10040 DATA 1,129,128,0,255,0,1,255,128
10050 DATA 1,24,128,2,60,64,2,36,64
10060 DATA 4,0,32,4,0,32,14,0,112,0
10100 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10110 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10120 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10130 DATA 0,0,0,0,0,0,0,0,0,255,255,255
10140 DATA 255,255,255,255,255,255,0
10200 DATA 0,0,0,0,0,45,0,27,3,0,0,8,0,127,0
10210 DATA 0,5,0,0,236,0,0,67,0,2,64,240
10220 DATA 7,32,112,7,32,56,15,145,56
10230 DATA 31,23,120,31,25,124, 95, 97, 254
10240 DATA 127,255,254,255,255,255,255
10250 DATA 255,255,0,0,0,0,0,0,0,0
```

Accesorio Acorn

El módulo de ampliación Plus 3 de Acorn Electron eleva el equipo al nivel requerido por usuarios serios de micros

Una de las críticas más serias que se le han hecho al Acorn Electron ha sido su falta de interfaces. Si bien la máquina utilizaba el veloz BASIC BBC, el hecho de que ésta sólo contara con un par de conectores para pantalla, un enchufe RF y una interface para cassette (no proporcionaba siquiera una puerta para palanca de mando) significaba que su potencial de ampliación estaba severamente limitado. Acorn siempre prometía módulos de ampliación que permitirían que los usuarios elevaran sus máquinas al nivel del BBC Micro, desde el punto de vista de interfaces disponibles, pero estos accesorios tardaban mucho en llegar.

El primer módulo de ampliación que salió al mercado fue el Plus 1, que ya hemos tenido ocasión de examinar, que contribuyó mucho a acallar las críticas que se le hacían al Electron. Pero aunque proporcionan muchas de las interfaces de que no disponía el Electron original, el Plus 1, sin embargo, no incluyó una interface para disco que permitiera el fácil acceso al almacenamiento masivo de datos.

Ahora esta situación ha cambiado gracias a la aparición del Plus 3, un sistema de disco "todo en uno" para el Electron. El módulo tiene forma de "L", al instalarse el sistema de archivo en discos a lo largo de la parte posterior del ordenador, con la unidad de disco, más corta, colocada sobre el lado derecho; esta última, lamentablemente, tapa la fuente de alimentación eléctrica del Electron. El Plus 3, en consecuencia, viene con su propia fuente de alimentación eléctrica, de 36 V, que se enchufa en el lado izquierdo de la unidad y alimenta tanto a la unidad de disco como al Electron. Al igual que el Plus 1, el Plus 3 se desliza en el conector marginal de la parte trasera del ordenador y, para evitar que la máquina se mueva y pueda romper el conector marginal, el módulo se fija firmemente en la base de ésta mediante dos tornillos. Una vez instalado, el Electron y el Plus 3 conforman una sólida unidad, muy difícil de romper. A diferencia del Plus 1, el nuevo modelo posee su propio conector marginal instalado atrás, para permitir la adición del Plus 1 y de las unidades que pudieran aparecer en el futuro.

Lo que tal vez resulte sorprendente es que Acorn haya optado por dotar al Plus 3 de unidades de disco Sony de 3 ½ pulgadas, en vez del formato usual de unidades de disco de 5 ¼ pulgadas. Situado directamente detrás de la unidad de disco hay, asimismo, un segundo conector marginal que permite la adición de una segunda unidad de disco, la cual según Acorn utilizará no sólo unidades de 3½

Chris Stevens

Bloques armónicos
El accesorio Plus 3 de Ac
para el Electron está diseñ
para acomodar también a
1. La provisión de estas
interfaces permite que el
del Electron mejore su
ordenador, alcanzando un
especificaciones que se
aproximan a las del BBC M
(con unidad de disco) a un
precio muy razonable

pulgadas, sino también de 5 ¼ pulgadas. Si el usuario desea agregar una unidad de disco de 5 ¼ pulgadas para el BBC Micro, Acorn ofrece una interface accesoria para poder instalar la unidad en el conector marginal.

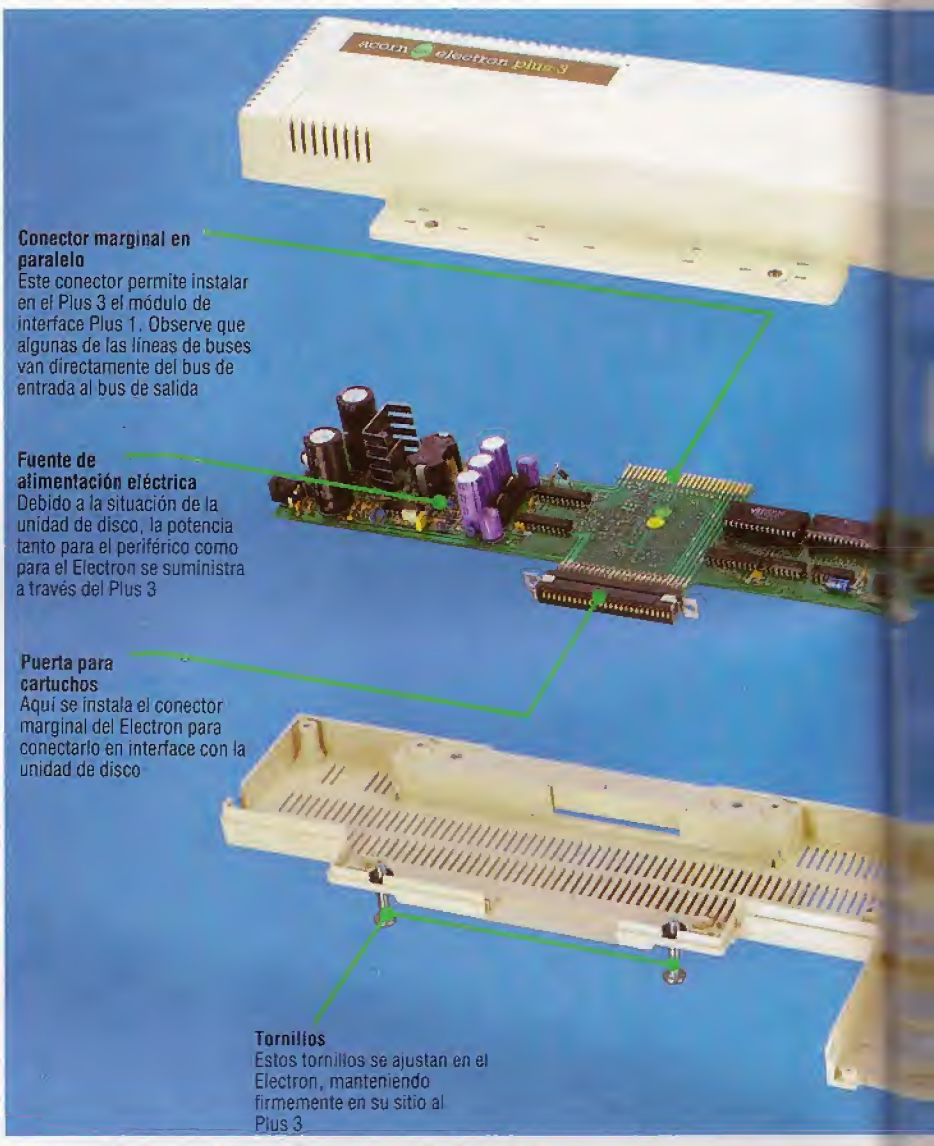
En el interior del dispositivo, el Plus 3 se divide en cuatro partes. Comenzando por la izquierda, está el sistema de circuitos de la fuente de alimentación eléctrica, caracterizado por sus grandes condensadores. A continuación se hallan los buses que salen del Electron, algunos de los cuales conducen a la unidad de disco, y otros vuelven directamente al conector marginal externo que se conecta con el Plus 1. A continuación viene el sistema de circuitos del sistema de archivo en disco. Aquí los chips contienen las instrucciones extras basadas en ROM que permiten que uno le "hable" a la unidad de disco y los chips de gestión de los discos. Por último, a la derecha está la propia unidad de disco de doble cara, revestida en metal. Esta carcasa tiene un doble papel. Básicamente cumple el papel de disipador, para impedir que la temperatura del interior del módulo se eleve excesivamente, pero también actúa como caja de Faraday, impidiendo que los campos magnéticos dispersos interfieran en el sistema.

Los discos Sony están alojados en un plástico rígido, con una cubierta protectora en la ventana de lectura/escritura. Al formatearse, el disco se divide en 80 pistas de 16 sectores. Puesto que cada sector puede contener hasta 256 bytes de información, la capacidad total de un disco de una sola cara es de 320 Kbytes.

Cuando se instala el Plus 3 y se enciende la máquina, junto con el encabezamiento normal se visualiza el mensaje ACORN ADFS. Ésta es la sigla de Advanced Disk Filing System. Aunque el Electron DFS es muy similar en muchos sentidos al sistema del BBC Micro (muchas de las instrucciones son idénticas), existen sutiles diferencias en la operación del sistema. Con el Plus 3 se proporciona un disco Welcome que contiene programas de demostración similares a los de la cassette que viene con el Electron. Sin embargo, el disco también contiene numerosas utilidades de incalculable valor para la operación del sistema de archivo en disco. Entre las utilidades que se proporcionan se incluyen BACKUP, que permite copiar discos enteros, DIRCOPY, que permite copiar de un disco a otro sólo archivos específicos, BUILD, que permite incorporar al disco rutinas BOOT, y DUMP, que visualiza el contenido de un archivo en formatos hexadecimal y ASCII.

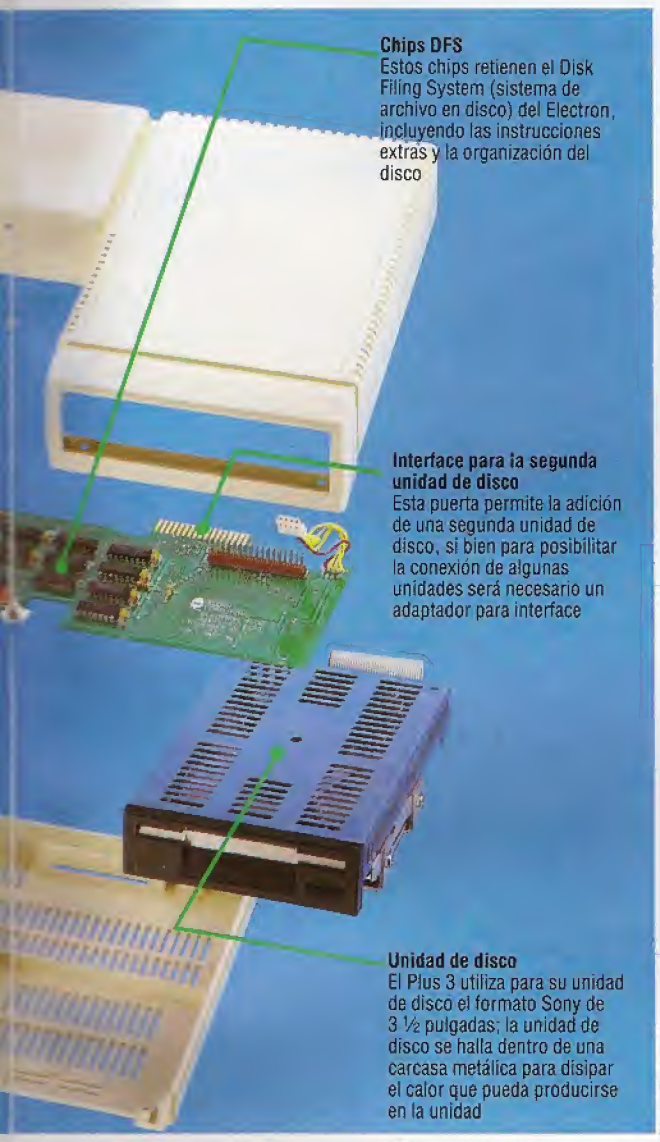
Al igual que el BBC Micro DFS, las instrucciones de disco del Electron van precedidas por la instrucción *, de modo que instrucciones del OS como *LOAD, *RUN y *DELETE les resultarán familiares a quienes hayan utilizado el sistema de archivo en disco del BBC Micro. La catalogación (CAT) del disco producirá una visualización idéntica a la del Micro, con el título del disco, el número de unidad, las opciones, el nombre del directorio actual y el nombre de la biblioteca actual con los archivos, listados debajo.

La diferencia fundamental entre los dos sistemas operativos es el método que emplean para cargar discos nuevos. Cuando se inserta el disco en la unidad del Plus 3, el usuario ha de informar al sistema operativo digitando la instrucción *MOUNT, que tiene el efecto de cargar en la memoria el Currently



Selected Directory (CSD) y la Currently Selected Library (CSL). Cuando el usuario desea cambiar de disco, debe entrar la instrucción *DISMOUNT. Esta instrucción cierra todos los archivos secuenciales que pudiesen estar abiertos e inhabilita el CSD y la CSL. Una vez cambiado el disco en la unidad, se vuelve a impartir la instrucción *MOUNT. Exceptuando la adición de estas instrucciones, el sistema operativo de disco parece ser idéntico al del BBC Micro.

El ADFS está organizado por una serie de jerarquías de archivo. Ello permite el acceso fácil y rápido a los archivos, gracias a una serie de nombres de ruta (*pathnames*). En el extremo superior de la jerarquía se halla el directorio seleccionado actualmente. Tras el encendido, éste es por defecto el directorio raíz $. Para ir del CSD a cualquier otro directorio disponible en el disco actual, el usuario simplemente digita *DIR seguido del nombre de ruta elegido. Debajo del CSD puede haber una cantidad de archivos, o una cantidad de subdirectorios conocidos como *archivos de biblioteca*. Dado que el CSD sólo puede retener hasta 47 archivos al mismo tiempo, usted debe organizarlos en bibliotecas si es que desea retener en el disco una cantidad muy elevada. Estos subdirectorios pueden retener una cantidad de archivos que quizá no pudieran in-



Chips DFS
Estos chips retienen el Disk Filing System (sistema de archivo en disco) del Electron, incluyendo las instrucciones extras y la organización del disco

Interface para la segunda unidad de disco
Esta puerta permite la adición de una segunda unidad de disco, si bien para posibilitar la conexión de algunas unidades será necesario un adaptador para interface

Unidad de disco
El Plus 3 utiliza para su unidad de disco el formato Sony de 3 ½ pulgadas; la unidad de disco se halla dentro de una carcasa metálica para disipar el calor que pueda producirse en la unidad

Chris Stevens

Pantallas del Acorn ADFS
El Acorn Advanced Disk Filing System para el Electron es similar al del BBC Micro. La pantalla superior muestra la visualización en respuesta a la instrucción *CAT. *INFO visualiza el nombre del archivo, sus permisiones de acceso, carga, ejecución y direcciones de disco, y la longitud del archivo

Ian McKinnell

ACORN ELECTRON PLUS 3

DIMENSIONES

365 x 200 x 50 mm (unidad de disco);
365 x 90 x 50 mm (interface)

CAPACIDAD

320 K para un disco de una sola cara

INTERFACE

Enchufe para fuente de alimentación eléctrica, bus de entrada tipo cartucho, bus de salida de conector marginal

DOCUMENTACION

El manual que se entrega con el Plus 3 es exhaustivo y le ofrece al principiante un enfoque paso a paso para la utilización del dispositivo. No obstante, sería necesario corregir algunos detalles ínfimos

VENTAJAS

Por su precio, la interface es una buena inversión y el usuario obtiene al mismo tiempo un sistema de archivo en disco y una unidad de disco

DESVENTAJAS

Por el momento hay muy poco software disponible para la máquina. Además, para hacer un uso completo de la unidad de disco se necesitan extras tales como una interface para impresora, que sólo está disponible en el Plus 1

cluirse en el directorio CSD principal; por consiguiente, para acceder a un archivo llamado Pepe en un disco con un cierto número de directorios con bibliotecas, el formato sería $.library1.pepe. Aunque de entrada este método para acceder a un archivo puede parecer muy complicado, tiene la ventaja de permitir que cada una de las bibliotecas disponibles en el disco tenga un archivo llamado Pepe. Siempre que el usuario permanezca en la biblioteca adecuada, al volver a guardar (SAVE) el archivo éste no machacará los otros archivos Pepe retenidos en el disco.

Al igual que en el BBC Micro, en el ADFS del Electron se proporcionan numerosas utilidades de gran valor práctico. Hay, por ejemplo, una serie de instrucciones que permiten cambiar el nombre de discos, directorios y archivos. Hay disponibles, asimismo, varias instrucciones para manipular las permisiones de acceso a los archivos. Ello se realiza mediante la instrucción *ACCESS, que limita el acceso a un archivo. Por ejemplo, generando la instrucción *ACCESS file1 RL, se "bloquea" el fichero, evitando su borrado accidental, pero permitiendo su lectura. Por otro lado, impartiendo la instrucción *ACCESS file1 WR se puede leer el archivo, o escribir en él, con entera libertad. Para impedir que alguien copie del disco un archivo en código máquina, colocándole al archivo el sufijo E se limitarán la cantidad de instrucciones que se pueden utilizar en el archivo para ejecutarlo (*RUN) y también se limitarán los diversos métodos para borrarlo.

El manual que se incluye con el Plus 3 es muy amplio, incluyendo un curso de enseñanza completo, y responde al nivel de documentación propio de Acorn. Sin embargo, el manual tiene algunos ligeros descuidos que, aun siendo menores, resultan algo molestos. Por ejemplo, en la página 18 de la guía se mencionan las abreviaturas CSD y CSL, pero éstas no se explican hasta las páginas 22 y 26, respectivamente.

Desde que se lanzó el Electron, por lo general se lo ha considerado como el pariente pobre, de la familia Acorn, del BBC Modelo B Micro. La carencia de interface, en especial la omisión de una puerta para unidad de disco, llevó a muchos a la conclusión de que el Electron era poco más que una máquina de juegos que no se podía considerar como una buena inversión para aplicaciones serias. Sin embargo, con la reducción del precio del Electron y la aparición de las interfaces Plus 1 y Plus 3, ahora la máquina aparece como una propuesta mucho más interesante. Ahora el Electron se ha convertido en una fuerza a tener en cuenta incluso por los usuarios más exigentes.

Preámbulo

Iniciamos el diseño y construcción de una pequeña máquina capaz de asir y desplazar objetos

El análisis de las opciones de diseño por lo general comienza con la definición de la tarea que deseamos llevar a cabo. En nuestro proyecto, la tarea nominal es que la máquina sea capaz de desplazarse desde un punto de partida, recoger algo (supongamos, un objeto del tamaño y peso de un carrete de hilo de coser) y colocarlo en una caja, antes de retornar a su punto de partida, con un margen de 2,5 mm. Supondremos que todas las posiciones se conocen de antemano. Una extrapolación industrial de esta tarea es una máquina que empaquete artículos a medida que van llegando a través de una línea de producción.

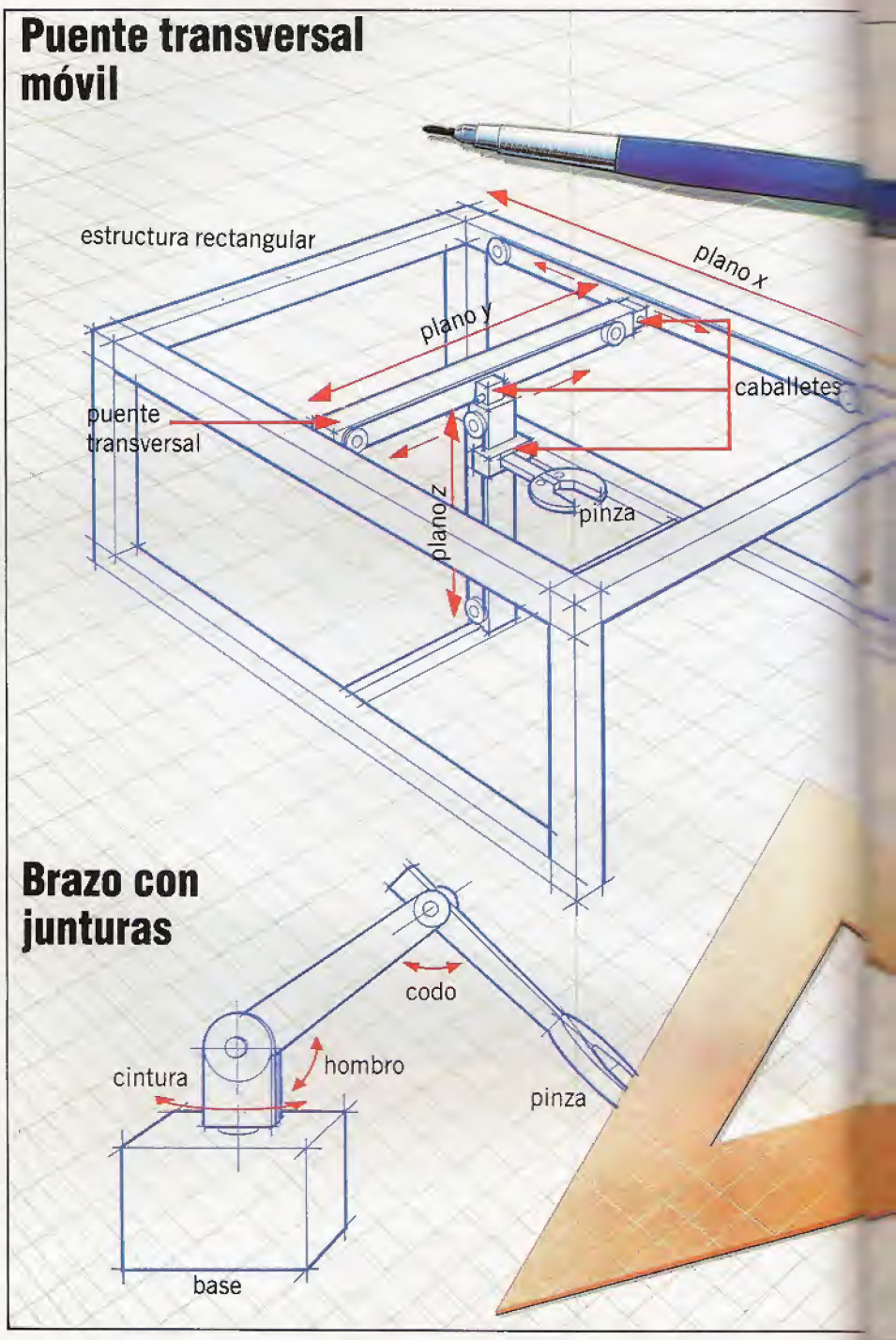
¿Qué clase de máquina se precisaría para llevar a cabo esta labor? Evidentemente necesitaría varios motores. Estos motores podrían ser hidráulicos o neumáticos, pero sólo consideraremos un sistema de motores eléctricos.

De los tres tipos principales de motor eléctrico, el más simple es el tipo CD normal. Aunque tales motores son económicos, es difícil controlarlos con precisión desde un dispositivo digital, tal como un ordenador personal. Para obtener el grado de precisión requerido utilizando este tipo de motor, habríamos de añadirle codificadores de eje a los husillos del motor. Cada motor requeriría entonces una lógica decodificadora, mediante hardware o software, para permitir un control posicional exacto.

Los motores paso a paso y los servocontroladores de CD son mucho más fáciles de controlar por ordenador. Sin embargo, los motores paso a paso, como ya hemos visto, son caros, requieren toda clase de circuitos activadores y suelen ser demasiado grandes para nuestros fines. Para un sistema a pequeña escala como el proyectado, los servomotores ofrecen dos ventajas fundamentales: dado que se producen en forma masiva para el mercado del modelismo controlado por radio, se pueden conseguir con facilidad y son relativamente económicos.

En nuestro apartado ya hemos visto cómo trabaja un servomotor; en resumidas cuentas, un sistema de servomotor se compone de un pequeño motor CD de 5 V, controlado por un bucle de realimentación digital. La posición angular del husillo del motor se controla mediante un pequeño potenciómetro y se compara con la señal de instrucción del chip incorporado. La señal de instrucción es un impulso de 5 V de entre uno y dos milisegundos de duración, determinando la longitud del impulso el ángulo que abarca el husillo del motor. Los impulsos de instrucción deben repetirse una vez cada 50 Hz (ciclos por segundo).

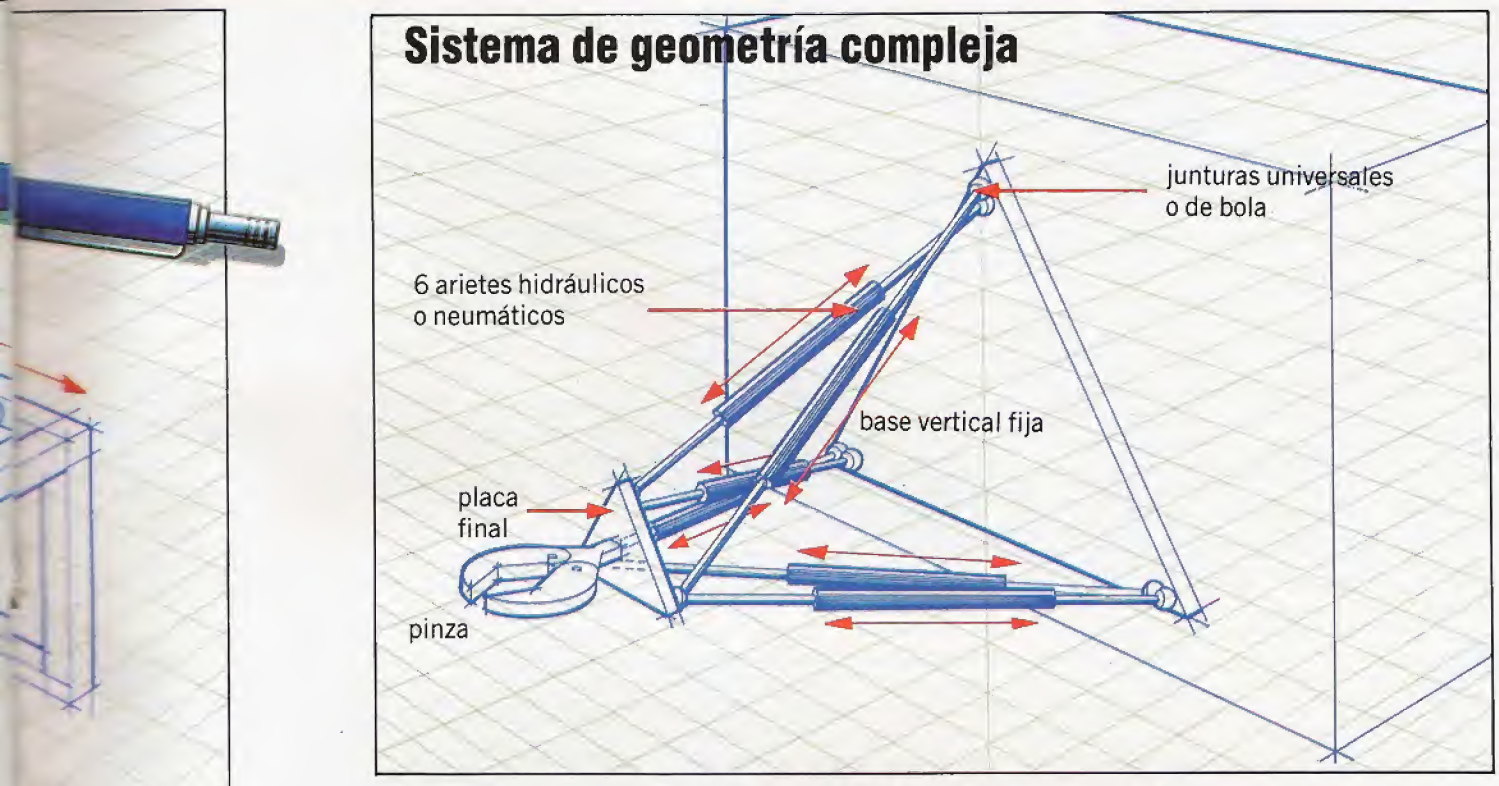
En nuestra serie también hemos descrito el software activador para servomotores tanto simples como múltiples. El programa es activado por interrupciones, lo que permite que el software en BASIC se ejecute como tarea de fondo y se interrumpa a intervalos regulares para enviar a los motores una nueva serie de impulsos.



Posibles prototipos
He aquí dos posibles diseños para máquinas que puedan recoger una bobina de hilo y colocarla en una caja o recipiente. El puente transversal móvil no puede realizar ciertas tareas más complejas de las que sí es capaz el brazo-robot. Ofrece la ventaja de una geometría de movimiento más simple; los motores del brazo-robot exigen un control mucho más complicado para obtener un movimiento lineal simple de la pinza. No obstante, el control por ordenador significa que estos cálculos complejos son fáciles de manejar

La geometría de la máquina

De inmediato surgen muchos diseños posibles para una máquina que lleve a cabo la tarea que hemos descrito, pero primero debemos considerar las limitaciones de diseño. La primera consideración es que para recoger un objeto como un carrete de hilo necesitamos un mecanismo de presión. Éste debe utilizar como mínimo un motor. En segundo lugar, la "pinza" se debe mover en tres planos. La máquina debe ser capaz de moverse al menos



Geometría compleja
El control por ordenador de motores eléctricos, hidráulicos y neumáticos facilita el desarrollo de máquinas de gran flexibilidad. La compleja interacción de los diversos motores que mueven las secciones de la máquina se puede manipular mediante software sofisticado, lo que, en cierta medida, queda oculto al usuario de la máquina. Piense en las complejas manipulaciones que permite este sistema de seis arietes, estando los seis arietes extensibles bajo el control de software desde un ordenador

10 cm de izquierda a derecha (el plano *x*), unos pocos centímetros hacia atrás y adelante (el plano *y*) y al menos 10 cm hacia arriba y abajo (el plano *z*). Cada plano de movimiento requerirá al menos un motor, y la pinza debe maniobrarse mediante una combinación de estos motores con una precisión de al menos 2,5 mm, para permitirle retornar a su punto de partida.

Asimismo, la pinza debe ser capaz de recoger un peso de al menos 20 g, y sus uñas deben ser capaces de abrirse al menos 3 cm para hacer lugar a un objeto del tamaño de una bobina de hilo.

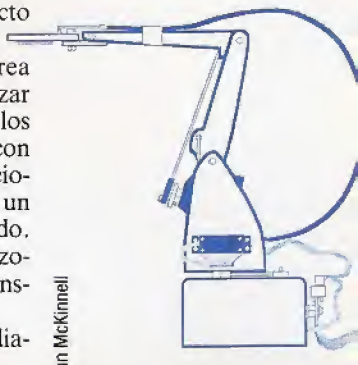
Un diseño que satisfaría estos requisitos sería un pequeño sistema de puente transversal basado en una estructura rectangular. El puente transversal y la pinza se podrían desplazar uniéndolas mediante caballetes a poleas de cable, que a su vez se podrían conectar a servomotores o motores paso a paso. Este sistema requeriría cuatro motores, tres para controlar el movimiento y uno para controlar la acción de prensión.

Una segunda posibilidad, y la que más se aproxima a la forma humana, es un brazo con junturas. La rotación de cada juntura se podría controlar mediante un solo motor. Los servomotores digitales son muy aptos para esta aplicación, dado que el ángulo del motor se dirige fácilmente mediante software para ordenador, permitiendo un control exacto de los miembros del brazo. El movimiento de la pinza mediante un brazo-robot requiere un control más complejo que efectuar el mismo movimiento mediante un puente transversal. Por ejemplo, para mover la pinza sólo en un plano (supongamos, el plano *x*) se requiere el control simultáneo de al menos dos motores del sistema de brazo con junturas, mientras que en el sistema de puente transversal este movimiento se podría conseguir utilizando sólo un motor. Sin embargo, dado que el sistema se controlará por ordenador, las secuencias de control complejas no ofrecen mayor problema: se pueden obtener mediante el uso de un software inteligente.

El brazo con junturas posee una ventaja respecto al puente transversal: es más flexible. Mientras que el puente transversal podría realizar la tarea específica que hemos esbozado, no podría realizar otras tareas sin una reacomodación física de los componentes del sistema. El sistema de brazo con junturas, sin embargo, podría efectuar modificaciones de la tarea establecida, tales como colocar un carrete de hilo en una caja que estuviera a su lado. En consecuencia, elegiremos el sistema de brazo-robot como base para nuestro proyecto de construcción.

En este primer capítulo proporcionamos un diagrama esquemático del diseño básico.

Color por números
En cada capítulo del proyecto iremos coloreando la sección pertinente de este esquema para indicar la parte del brazo-robot de la que nos estemos ocupando



Ian McKinnell

¿1+2=20?

En este capítulo analizaremos ciertos aspectos de las habilidades aritméticas del PASCAL

¿Cuánto es uno más dos? Sin más información, podríamos aventurar la respuesta: tres, por supuesto. El supuesto implícito es que estamos tratando con números naturales puros; pero supongamos que introducimos una moneda de 10 peniques y dos monedas de 5 peniques en una máquina tragaperras: el resultado sería 20 peniques (o, considerado desde un ángulo diferente, una taza de café y dos peniques de vuelta). Esta clase de clasificación de datos es fundamental para la descripción de cualquier problema, y el PASCAL nos ayuda a organizar nuestros datos y describirlos de una forma clara y lógica.

Como ya hemos visto, podemos crear categorías de datos nuevas y especiales, de modo que podamos pensar en la solución del problema y diseñar algoritmos basados en el proceso de cada objeto de datos de una forma apropiada a su tipo. Si inadvertidamente intentáramos hacer algo impropio, como leer del teclado un valor booleano, el compilador de PASCAL detectaría de inmediato este error lógico. Como comprenderá, ¡se pueden ahorrar muchísimas horas de frustrante trabajo de depuración si el compilador del lenguaje no lo deja siquiera ejecutar una sola instrucción hasta haber eliminado todos los errores del programa fuente! La incompatibilidad entre diferentes tipos de datos y la riqueza de las formas en las que podemos describirlos es una de las cualidades más valiosas del PASCAL. Lejos de sentirnos limitados en cuanto a lo que podemos hacer, con frecuencia desearemos aprovechar más las descripciones de datos fuertemente categorizadas en tipos del PASCAL e imponerles a las variables de forma deliberada algunas restricciones adicionales por nuestra propia cuenta.

Ahora que ya hemos utilizado todos los tipos de variables simples (aquellas que sólo pueden tener un valor), podemos resumir las reglas para manipularlas. Ninguna de las escalares es "compatible" con una variable de cualquier otro tipo escalar, si bien los dos tipos numéricos poseen muchas características en común y de vez en cuando se les permite "codearse" entre sí. Puesto que un número real se puede aproximar a un número entero, es posible la asignación de enteros a reales, pero jamás al contrario.

He aquí algunos ejemplos:

```
Program Compatibilidad (input,output);

VAR
   entA,
   entB :integer;
   Xreal,
   Yreal :real;

BEGIN
   read (entA, Xreal); {leer un entero, luego cualquier
   número legal}
Yreal:=entA; {real:=entero es OK}
entB:=Xreal;{**ERROR: ilegal**}
{..etc.
```

Las operaciones aritméticas sólo se definen aplicadas a los tipos numéricos, lo que no es del todo sorprendente. Tanto para los enteros como para los reales, se pueden utilizar los cuatro "operadores" simbólicos usuales:

+ suma
− resta
* multiplicación
/ división con punto flotante

En este contexto, son operadores *diádicos* o binarios, dado que siempre exigen dos "operandos" de cualquiera de los tipos numéricos. Cuando uno de los operandos es real, el resultado de la expresión es real: de modo que 2+2.0 es 4.0 (no 4). En el caso del signo de división, la expresión da un valor numérico real cuando ambos operandos son enteros: 3/5 es 0.6, 8/4 es 2.0.

Cuando dividimos valores integrales, con frecuencia una respuesta "real" no tiene sentido. Doce chocolatinas divididas entre diez personas da una para cada una y dos chocolatinas sobrantes, por ejemplo. El resultado entero y el resto se pueden obtener mediante los dos operadores de división entre enteros, DIV y MOD (que en PASCAL son ambos palabras reservadas): 15 DIV 5 = 3 y 15 MOD 5 = 0; 31 DIV 7=4 y 31 MOD 7=3. Tenga cuidado si es posible que sus datos sean negativos, dado que la división entre enteros no está definida para valores negativos del denominador. Con ambos tipos de división, de reales y de enteros, todo intento de división por cero resultará en un error en tiempo de ejecución: al menos hasta que alguien invente alguna forma de calcular el infinito.

En una expresión, por supuesto, las operaciones de multiplicación y división se evalúan antes que la suma y la resta. Debe utilizarse la notación habitual de paréntesis para evitar esta "precedencia" lógica. Por ejemplo: (8+4)DIV 2=6, pero 8+4 DIV 2=10.

Funciones de transferencia

Aunque no podemos efectuar una asignación directa de un valor real a un entero, el PASCAL proporciona "funciones de transferencia" muy útiles, con los identificadores trunc y round, que se pueden emplear para llevar a cabo esta asignación. Trunc simplemente trunca un número real, independientemente de su parte fraccional, de modo que trunc (3.999)=3 y trunc (−123.456)=−123. La función round realiza un redondeo inteligente, a cero si la parte fraccional es inferior a 0.5 y alejándose de cero, en caso contrario: por consiguiente, round (1234.5)=1235 y round (−0.49237)=0. Naturalmente, se producirá un error si el argumento real de

alguna de estas funciones diera un resultado entero que cayera fuera de la escala del "tipo" *integer* (entero) (de −MaxInt a MaxInt), de modo que compruébelo siempre primero con una sentencia IF. Asimismo, el argumento *debe* ser real, no entero (éstos, al fin y al cabo, ya están truncados y redondeados).

Hay un bonito detalle que no les resultará familiar a los programadores que han desfallecido ante expresiones que utilicen la función INT del BASIC. El PASCAL posee una función predefinida que devuelve un resultado booleano: *false* si su argumento entero es un número par y *true* si es impar. El identificador utilizado se denomina odd.

```
IF odd(N)
  THEN
    WriteLn('Es impar!')
  ELSE
    WriteLn('Es par!')
```

o:

```
IF odd(N)THEN
  IF N MOD 2=0 THEN
    WriteLn ('Atrapa al compilador!')
```

La tabla muestra todas las funciones aritméticas del PASCAL. Todas pueden tomar cualquier argumento numérico simple, real o entero, y las dos primeras, abs y sqr, devolverán un valor de tipo compatible con su argumento. Por consiguiente: abs (−19.372) es 19.372, y abs(255) es 255. Las otras funciones, sin embargo, *siempre* darán un resultado real, de modo que sqrt(16) es 4.0, no 4. Ello se debe a que los algoritmos utilizados para evaluarlas están basados en el sumatorio de una serie de términos, todos los cuales son fraccionales.

Aquí encontramos una importante lección de estilo con respecto a la comparación de reales. Observe que al referirnos a los resultados reales de las funciones aritméticas hemos utilizado la palabra "es" en lugar de un signo de igualdad. Lo que queremos expresar es "es lo mismo que" y no "es igual a", porque *nunca* es seguro comparar valores reales a un grado exacto de igualdad. El más mínimo error de cálculo significará que dos reales nominalmente "iguales" puedan en realidad diferir en, supongamos, 1.0E-27. Es insignificante, por supuesto, pero el ordenador no lo sabe. En vez de utilizar IF X=Y THEN..., podemos utilizar la función predefinida abs del PASCAL:

```
IF abs(X−Y) < Insignificante THEN {etc.}
```

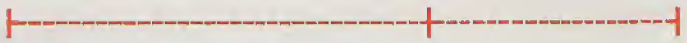
Aquí comprobamos el caso en el cual la diferencia entre X e Y es sumamente pequeña. Sería muy útil especificar en una definición CONST al comienzo del programa exactamente qué consideramos insignificante. Los logaritmos se dan en su base natural (e), no 10, y exp eleva e a la potencia de su argumento (es decir, se "exponencia"). El PASCAL no posee un verdadero operador de exponenciación, lo que obedeció a una decisión de diseño que tomó Wirth de forma deliberada. En muchas ocasiones usted habrá visto en programas en BASIC cosas tan tontas como:

```
500 LET D=B^2+4*A*C
```

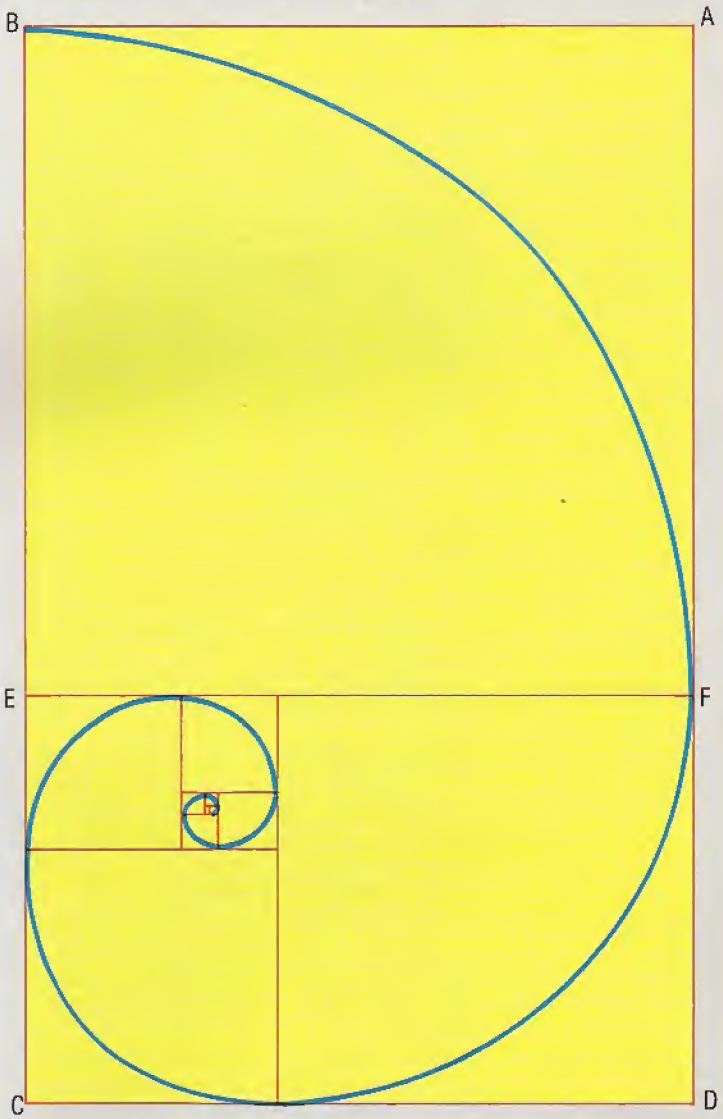
que utiliza la forma más lenta y menos exacta posible de calcular el cuadrado de un número. B*B es muchísimo mejor, por supuesto. Observe que sqr(N) del PASCAL devuelve el cuadrado de N direc-

La sección dorada

Existe un solo punto en la línea que la divida en dos partes de modo tal que la proporción de la longitud de la sección más pequeña respecto a la grande sea la misma que la proporción de la mayor respecto a la longitud de la línea. Esta proporción se denomina *sectio aurea* o *sección dorada* y es un número irracional con una cantidad infinita de dígitos



La sección dorada la utilizaban los artistas y arquitectos griegos (especialmente para el diseño del Partenón) y suele revivirse en las teorías de proporción. Sin embargo, algunas de sus manifestaciones más familiares se encuentran en la naturaleza. La espiral de "concha marina" se basa en ella, dado que sus vectores de radio, separados por 90°, se hallan en esta proporción de secciones



Dentro del rectángulo ABCD se puede trazar un cuadrado ABEF, de modo que se forme un rectángulo (EFDC) con la misma proporción de lados que ABCD. El proceso se puede continuar *ad infinitum* y los vértices de los cuadrados son puntos situados en la "espiral de sección dorada"

tamente como un entero verdadero (asumiendo que N es un entero); sqr(X/3) daría un real. Observe, asimismo, otro hábito del BASIC que habrá de desterrar: sqrt(it), *no* sqr(it) si desea la raíz cuadrada de un número.

Tipos de subrango

Todos los tipos escalares poseen un rango de valores definido y ordenado, y se los puede utilizar como un tipo "anfitrión" del cual se pueden derivar nuevos tipos de subrangos. Éstos se componen, como sugiere su nombre, de una porción contigua del tipo anfitrión. Los límites inferior y superior necesarios para delimitar el subrango se listan entre el signo de "elipsis" (..) en la definición del identificador de tipo de subrango; por ejemplo:

```
TYPE
  byte=0..255 {subrango de integer}
  alpha='A'..'z';{subrango de char}
  baraja=(trebol, diamante, corazon, pica);{un tipo
        nuevo}
  mayor=corazon..pica; {subrango de baraja}
```

Aparte de las ventajas que hemos analizado antes, un buen compilador optimizador de PASCAL almacenará variables de tipo byte, por ejemplo, en ocho bits en vez de los 32 posibles (si se emplea la representación de entero de cuatro bytes). Por consiguiente, una gran matriz de este tipo ocupará sólo el 25 % de la memoria que se utilizaría con elementos enteros no cualificados. Observe que el tipo mayor tiene dos valores posibles, algo así como pensar en la definición de valores booleanos como:

```
TYPE
  booleano=(false, true);
```

Todos los tipos de subrango heredan de su tipo anfitrión tanto la clasificación de datos como las operaciones en él definidas.

La definición anterior de *alpha* podría permitir, por supuesto, otras cosas además de letras del alfabeto. En particular, el juego de caracteres ASCII posee otros seis símbolos en el espacio entre "Z" y "a", incluyendo los símbolos de los corchetes ('['y']'). Éstos también figuran entre los 23 símbolos reservados del PASCAL, y se utilizan para delimitar los componentes de algunos tipos de datos estructurados, como en muchos lenguajes. Originalmente el BASIC también empleaba estos símbolos para encerrar los índices de matriz, pero la sintaxis se cambió por los paréntesis debido a que los ordenadores antiguos a veces poseían juegos de caracteres "medio ASCII" que no incluían letras en minúscula, corchetes, las llaves que utiliza el PASCAL para encerrar los comentarios, y algunos otros símbolos. El Apple II sin modificar también adolecía de esta restricción.

El lenguaje PASCAL proporciona las siguientes alternativas opcionales para los casos en que exista este problema: se permite (.y.) en lugar de [y], y (*y*) se pueden utilizar en lugar de {y}. La segunda alternativa, para delimitadores de comentarios, es bastante común; no obstante, es probable que sólo los compiladores autorizados por la ISO soporten las alternativas de los corchetes. Se puede realizar cualquiera de las sustituciones o ambas, de modo que: (*este es un comentario legal}.

Función	Valor devuelto
abs (K)	valor absoluto de K
sqr (K)	cuadrado de K
sqrt (K)	raíz cuadrada de K
sin (A)	seno de A radianes
cos (A)	coseno de A radianes
arctan (T)	ángulo (en radianes) con tangente T
ln (K)	logaritmo natural de K
exp (L)	*e* elevado a la potencia L ("anti-Ln"de L)

Banda de oro

El programa Dorada, que ofrecemos aquí, genera términos Fibonacci y los imprime junto con su proporción. Utilizando el programa, se puede ver que una de las propiedades interesantes de la serie de Fibonacci (1,1,2,3,5,8,13, etc.) es que la proporción de cada par sucesivo de términos se aproxima cada vez más al valor de la sección dorada. El programa demuestra algunos de los principios que ya hemos analizado en el curso, incluyendo nuevos ejemplos de la estructura REPEAT del PASCAL. A modo de ejercicio, trate de alterar la condición para la finalización del bucle de modo que se detenga cuando el siguiente término Fibonacci a calcular exceda el valor MaxInt

```
PROGRAM      Dorada      (output);

CONST
        Epsilon          =1.0E-08;

TYPE
        Fibonacci        =1..MaxInt;

VAR
        primero,
        segundo,
        siguiente    :Fibonacci;
        proporción,
        Oro          :real;
        contador     :integer;
BEGIN
  WriteLn ('La seccion dorada' : 30);
  WriteLn;
  WriteLn ('Series Fibonacci :');
  primero := 1;      {por definicion}
  segundo :=1;
  proporcion := primero/segundo;
  contador :=0;

  REPEAT
    IF contador MOD 10 = 0 THEN
      BEGIN {Cabecera cada 10 lineas}
        WriteLn;
        WriteLn ('Primero' : 10,
          'Segundo' : 10, 'Proporcion' : 14);
        WriteLn
      END;

    WriteLn (primero : 10, segundo :10,
          proporcion : 16 : 8 );
    contador := contador+1;
    Oro := proporcion;   {recordar antiguo GS}
    siguiente := primero + segundo;
    primero := segundo;      {ir avanzando hacia ..}
    segundo := siguiente;   {arriba de la serie}
    proporcion :=primero / segundo

  UNTIL abs (proporcion - Oro ) < Epsilon;

  WriteLn;
  WriteLn ('El termino medio dorado es : ',
        100*proporcion : 10 : 5, '%' )
  END.
```

"Interrumpimos este programa para..."

Con señales denominadas "interrupciones" y "eventos" el OS gestiona los recursos de proceso de un ordenador

Cuando se ejecuta un programa en BASIC con un BBC Micro, es fácil creer que todas las energías de la máquina están concentradas en esa ejecución. Pero no es así; constantemente se están realizando otras tareas como la lectura del teclado por parte del OS. Este aparente "tiempo compartido" de las potencialidades del ordenador es posible gracias al empleo de las *interrupciones*. Una interrupción es, en esencia, una señal dirigida a la CPU que la instruye para que detenga lo que esté haciendo en ese momento y realice otra tarea. Una vez realizada esta última tarea, la CPU vuelve a su trabajo original y continúa con él como si nada hubiera ocurrido. Las interrupciones en el BBC Micro se encargan de actividades tan dispares como la lectura del teclado, el parpadeo de colores, el proceso de la instrucción ENVELOPE y las lecturas del convertidor A/D. Ahondemos un poco más en cómo realiza esto la máquina.

Las interrupciones en el BBC Micro se generan mediante un buen número de dispositivos hardware incorporados al sistema del ordenador; entre ellos se encuentra el chip de Interface Serial, los VIA del sistema y del usuario (VIA: adaptador de interface versátil), un equipo físico Econet e Interface de Discos. Estos dispositivos originan dos tipos de interrupciones en el procesador 6502. Ambos *causan* que el sistema interrumpa lo que está haciendo, aunque se puede instruir a la CPU para que ignore uno de los dos tipos si se desea. Ésas son llamadas *peticiones de interrupción enmascarables*, o bien IRQ (*maskable interrupt requests*). El otro tipo de interrupción no será ignorado nunca por la CPU, convirtiéndose en una interrupción absolutamente prioritaria. Este tipo es la llamada *interrupción no enmascarable*, o bien NMI (*non-maskable interrupt*).

Las NMI en el BBC se reservan para dispositivos que necesitan una atención inmediata de la CPU, mientras que las IRQ son generadas por los dispositivos que solicitan la atención de la CPU, pero pueden esperarse un poco. Los únicos dos dispositivos que usan las NMI son el hardware de la red Econet y el sistema de ficheros en disco. Al recibir una NMI, el OS emplea una pequeña rutina en código máquina situada en la página &0D. Ésta es una de las razones por las que la página &0D queda a libre disposición del usuario una vez ajustada la interface para discos. A causa de estas propiedades especiales de la NMI, no la vamos a considerar aquí. Al programador se le desaconseja su uso en concreto por la misma Acorn.

Nos quedamos, pues, con las IRQ. Las interrupciones enmascarables del procesador 6502 pueden quedar desactivadas por medio de la instrucción SEI desde programas de ensamblador; y reactivadas por medio de la instrucción CLI. Cuando es recibida una IRQ por el 6502, se actúa según la fuente de la interrupción. El 6502 descubre el origen de la interrupción "preguntando" a cada posible fuente de interrupción si fue la causa de la interrupción. Tan pronto como se ha encontrado cuál es la fuente, la interrupción es manejable y el dispositivo que la generó es instruido para que olvide la interrupción hasta que ocurra otro evento que merezca el envío de una nueva interrupción al 6502. A esto se le llama *borrar* la interrupción. La rutina en código máquina que maneja la interrupción se denomina ISR (*interrupt service rutine:* rutina de servicio de interrupciones).

¿Qué hace una ISR?

Dado que el programa interrumpido no debe acusar cambio alguno en los valores existentes en los registros del 6502, la primera misión de una ISR es guardar los valores de los registros. Esto se hace en la pila del 6502 por medio de una serie de operaciones de carga (*push*). Así, una vez tratada la interrupción, estos valores pueden ser restaurados. La ISR debe posteriormente realizar todo lo necesario para servir la interrupción, borrar la condición de interrupción antes de abandonar la ISR y devolver el control al programa interrumpido. La razón por la que es tan importante que se borre la interrupción es obvia; si no se borrara, tan pronto como se devolviera el control al programa interrumpido el 6502 volvería a recibir una IRQ nuevamente desde la fuente, poniendo la CPU en un bucle sin salida.

Una vez sentados los principios generales del tratamiento de una interrupción, veamos el caso específico de un BBC Micro. Este ordenador se apoya en gran medida en las interrupciones, sirviéndose de ellas para un gran número de operaciones. Por esta razón, hay que tener cuidado de no desactivar las interrupciones demasiado tiempo por medio de SEI. Si cuidamos que el intervalo de tiempo de desactivación de las interrupciones sea inferior a unos pocos milisegundos, no ocurrirá nada especial. Pero un poco más de tiempo provocará un extraño comportamiento del OS. El siguiente programa demuestra lo que sucede en un BBC Micro cuando se desactivan las interrupciones durante diversos períodos de tiempo:

```
10 MODE 2
20 DIM C (100): REM establece el espacio para el
   cod. maq.'
30 FOR I%=0 TO 2 STEP 2
40 P%=C
```

```
50 [OPT I%
60 .code SEI/desactiva las interrupciones
70 RTS
80 ]:NEXT
90 COLOUR 14
100 PRINT "Hola!"
110 ENVELOPE 1,1,4,-4,3,10,20,20,127,0,0,
    -5,126,126
120 SOUND 1,1,160,200
130 TIME=0
140 REPEAT
150 PRINT TIME,I%
160 I%=I%+1
170 IF TIME > 100 THEN CALL code
180 UNTIL FALSE: REM no acaba jamas
```

Ejecute este programa y verá que en cuanto se desactivan las interrupciones llamando a la rutina en código máquina denominada code, la variable TIME no se altera y se detienen tanto los colores parpadeantes como el proceso ENVELOPE. Pero la máquina no se "cuelga", ya que la rutina continúa imprimiendo I% fielmente. Asegúrese de pulsar CTRL BREAK cuando haya finalizado la ejecución de este programa.

El vector IRQV1

Cuando el 6502 recibe una IRQ, el control se pasa a una ISR cuya dirección está contenida en un vector llamado ISQV1. Este vector está en la dirección &204 y &205. La rutina entrada por medio de este vector trata las interrupciones a través de tres dispositivos hardware: el chip de Interface Serial, el VIA del sistema y parte del VIA del usuario. Toman el orden de prioridad en que se han citado.

El chip de Interface Serial es una pieza del hardware que se encarga de la interface RS423 y de la interface de cintas. Las interrupciones llegadas por esta fuente conciernen, claro está, a estas interfaces. El chip es el dispositivo de máxima prioridad en el uso del vector.

Por este orden tenemos a continuación las interrupciones generadas por medio del VIA del sistema. Este dispositivo de tanta utilidad interrelaciona la CPU con otros muchos dispositivos de la máquina, y ofrece además algunas facilidades de temporizado. Una interrupción a través del VIA del sistema se obtiene al pulsar una tecla, que indica al OS que debe procesar la tecla de algún modo. Se produce también una interrupción por este chip cuando se envía un fragmento entero de información a la pantalla o al televisor. Tal interrupción se envía una vez cada 1/50 de segundo, y se conoce como *interrupción de sincronía vertical*.

El convertidor A/D actúa también en conjunción con el VIA del sistema, enviándonos una interrupción cada vez que se finaliza la conversión de una señal. Esta interrupción particular hace que el OS actualice los valores empleados para devolver un número a la función ADVAL. Otra última interrupción importante proporcionada por el VIA del sistema se genera una vez cada centésima de segundo. Hace que se incremente la variable TIME, que se actualice el "reloj de intervalos" (empleado para generar eventos), que se decremente el temporizador INKEY (que cronometra la demora en una instrucción INKEY) y que se procese parte de un ENVELOPE. También provoca que el OS continúe con el proceso de cualquier tecla pulsada. Otras interrupciones proporcionadas por medio del VIA del siste-

¡Camarero!
No es difícil imaginar la CPU del BBC Micro desempeñando un papel semejante al de un camarero en un restaurante muy concurrido. Lo mismo que el 6502 debe repartir su atención entre los periféricos, la transferencia de datos y la ejecución de programas, así un camarero debe atender a los clientes importantes (que solicitan preferencia sobre otros), limpiar las mesas y solucionar eventuales situaciones conflictivas.
Un camarero eficaz debe ser capaz de repartir su atención a cada una de estas tareas cuando se soliciten y además volver posteriormente a su punto de partida, la tarea menos prioritaria que dejó, de tal modo que el servicio de la cocina al comensal sea fluido. El 6502 obra del mismo modo, atendiendo a las "interrupciones" y a los "eventos" que el hardware o el control del software han generado. Una vez servida la interrupción, la CPU puede volver a su tarea anterior

Kevin Jones

ma se refieren al sistema sintetizador de la voz y al lápiz fotoeléctrico si se dispone de él. Existen las llamadas OSBYTE que nos permiten alterar la forma en que responde el OS a las interrupciones desde estas dos fuentes. Pero, como pronto veremos, no es aconsejable mezclarlas con el IRQV1.

La interrupción de menor prioridad tratada por este vector se pasa a una Rutina de Servicio de Interrupciones cuya entrada se produce mediante el VIA del usuario, programable por el usuario para obtener un buen abanico de interrupciones en la CPU. Normalmente tales interrupciones serán ignoradas por ésta.

El vector IRQV2

El vector IRQV2, contenido en las posiciones &206 y &207, puede ser alterado por el usuario para que apunte a una ISR creada por él. Hay que usar siempre este vector si deseamos añadir nuestras propias Rutinas de Servicio de Interrupciones, aunque es posible alterar el contenido de IRQV1 para apuntar a una nueva rutina. Este proceso de cambio del contenido de un vector que le obliga a apuntar a una nueva rutina es llamada *intercepción* del vector. Si se desea interceptar el vector IRQV2 para añadir rutinas de interrupción propias, se deben cumplir las siguientes condiciones a la entrada de la rutina apuntada por el vector:

- La CPU debe tener ya almacenado el contenido del registro A en la posición &FC y haber colocado el registro indicador de estado del procesador (PSR) en la pila.
- Los registros X e Y deben estar en el estado que tenían cuando ocurrió la interrupción.

El segundo punto quiere decir que se deben guardar los registros X e Y a la entrada de la propia ISR empleando las siguientes sentencias:

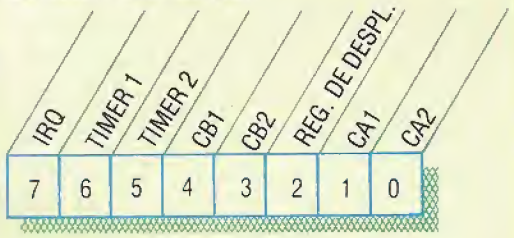
```
TXA:PHA
TYA:PHA
```

La rutina debe borrar la condición de interrupción. Esto depende de la fuente de la interrupción: si se trata del VIA del usuario, por lo general implica la lectura y escritura de ciertos registros del VIA del usuario. Antes de volver al programa interrumpido, hay que recordar restablecer los registros y emplear la instrucción RTI para abandonar la rutina de interrupción. Y finalmente se advertirá que aunque el registro A se almacena en la posición &FC al entrar a IRQV1, una interrupción ulterior puede causar su sobreescritura antes de acabar de procesar nuestra interrupción del usuario. Dos soluciones pueden darse: o bien ejecutar una instrucción SEI a la entrada de la rutina propia y una CLI antes de abandonarla, o bien llevar el registro A a la pila en la entrada de dicha rutina y restaurarlo antes de abandonarla. Si se hace esto último, debe también almacenar el valor del registro A en &FC antes de ejecutar la instrucción RTI. Los usuarios con poca experiencia del VIA del usuario harían bien en tomar todas las precauciones.

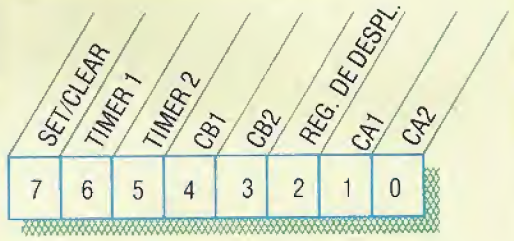
El BBC Micro emplea señales de "evento" como alternativa al uso de las interrupciones. Estas señales (que algunos consideran "interrupciones sin lágrimas") constituyen el tema que desarrollaremos en nuestro próximo capítulo.

Interrupciones del chip VIA

Registro del flag de interrupción



Registro de activación de la interrupción



Liz Dixon

Aunque el procesador 6502 del BBC MIcro tiene sólo dos líneas de interrupción (la IRQ y la NMI), los chips VIA de interface con periféricos pueden ser programados de tal modo que muchos dispositivos puedan compartir la línea de interrupción IRQ para el procesador. Cada chip VIA tiene dos puertas de E/S, cada una de las cuales tiene ocho líneas de datos y un par de líneas de señal. En la puerta A, son las denominadas CA1 y CA2, y en la puerta B, son CB1 y CB2. Estas líneas son a menudo empleadas para el apretón de manos, es decir, para la transferencia de datos sincronizada entre el micro y un dispositivo periférico, pero pueden también ser programadas para que actúen como líneas de interrupción. Además de las cuatro posibles líneas de interrupción que entran en cada VIA hay una interrupción interna que puede ser generada por el registro de desplazamiento del chip. Este registro se usa para la transmisión en serie por medio del CB2 en respuesta al impulso de una fuente externa sobre CB1. Finalmente hay también dos relojes de intervalos de 16 bits que pueden también generar interrupciones. Todas ellas deben compartir una única línea IRQ hacia el procesador. Cuando una interrupción se genera, el procesador puede que desee conocer la fuente de la interrupción. Esto es posible interrogando el registro del flag de interrupción en el VIA. Cada bit de este registro corresponde a una de las posibles fuentes y se pone a uno al solicitar la fuente una interrupción. El bit 7 es una excepción, pues se pone a uno si cualquier dispositivo pide una interrupción. Y es que el bit 7 es una línea IRQ. Pueden existir ocasiones en que deseemos detener un dispositivo en su interrupción mientras el procesador sirve otra diferente. Lo podemos conseguir programando el IER (registro de activación de la interrupción). Éste tiene ocho bits que corresponden a los mismos dispositivos que los bits del anterior IFR (registro del flag de interrupción). Si se pone un bit a uno se reconocerán las interrupciones del correspondiente dispositivo; si el bit está a cero éstas son ignoradas

Fortaleza espacial

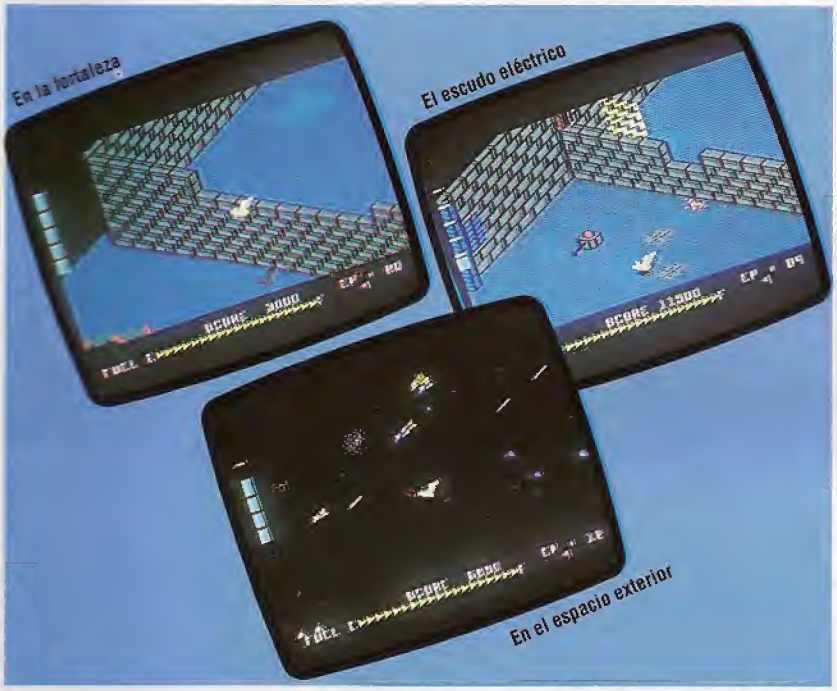
"Zaxxon" es un programa con excelentes gráficos tridimensionales, creado para las salas recreativas, que ahora está disponible para micros personales

La última frontera
Aquí vemos tres momentos de *Zaxxon*. El jugador inicia el juego pilotando una nave espacial hacia la primera de las dos fortalezas. La nave se debe maniobrar en el ángulo y la altitud correctas para poder atravesar la pared. En el interior de la fortaleza propiamente dicha hay otra pared, que está protegida por un escudo eléctrico. Se pueden conseguir puntos extras destruyendo las torretas con cañones. Una vez salvada la primera fortaleza, la nave sale al espacio, donde debe disparar contra gran número de naves enemigas que pasan junto a ella. Por último, tras superar las defensas de la segunda fortaleza, la nave se enfrenta directamente con el robot Zaxxon

El objetivo de *Zaxxon* es pilotar una nave espacial sorteando numerosos obstáculos en un intento por destruir al robot Zaxxon. Al principio su nave se halla en la inmensidad del espacio; finalmente aparece delante de usted una "fortaleza espacial", rodeada por una alta muralla. Para poder seguir adelante, debe pasar a través de un agujero que hay en esta barrera, antes de enfrentarse al primero de los obstáculos que encontrará en su camino. Éstos incluyen misiles, pantallas de radar y depósitos de combustible, todos en realistas gráficos tridimensionales.

Los depósitos de combustible son sumamente importantes puesto que su destrucción supone un aumento del combustible de su nave espacial.

El diseño de los gráficos es de gran belleza. Se



Ian McKinnell

utilizan tres sprites diferentes para representar la nave del jugador: uno para los picados, otro para los ascensos y el tercero para el vuelo normal. A medida que la nave se eleva o desciende en picado, el tamaño de estos sprites aumenta y disminuye de una forma muy convincente. Los obstáculos de tierra, asimismo, están bien dibujados y la perspectiva le otorga cierto nivel de realismo al juego. Un detalle especialmente atractivo es la sombra de la nave espacial; ésta va creciendo a medida que la nave pierde altura y pasa a través de los objetos que se hallan en tierra.

Si durante esta fase del juego su nave es destruida por el fuego enemigo, usted debe regresar al comienzo, pero al adquirir cierta práctica descubrirá que es posible sortear la mayoría de los peligros. El esfuerzo que supone destruir las pantallas de radar vale la pena, porque éstas, de dejarse solas, hacen que los misiles de desplazamiento horizontal se dirijan hacia usted. Antes de poder destruir un radar, se requieren tres impactos directos. En todo momento se debe tener un gran cuidado en conducir la nave a la altura correcta: es muy fácil leer de forma errónea el altímetro y volar directamente hacia un misil que se acerque. Finalmente el jugador habrá de enfrentarse con un campo de fuerza eléctrica: debe ascender muy alto para superarlo y luego descender en picado hasta un nivel más bajo, donde podrá destruir los cazas enemigos que lo esperan en tierra. Es aconsejable librarse de la mayor cantidad posible de éstos, ya que todo adversario eliminado en esta etapa desaparece definitivamente del juego.

Después de sortear con éxito las defensas de la fortaleza, su nave vuelve a encaminarse hacia el espacio, donde se enfrenta a un alto número de naves enemigas. En este punto, el usuario se enfrenta con el problema de jugar a un juego tridimensional en una pantalla bidimensional. Para atacar las naves enemigas es necesario ascender o descender hasta el nivel correcto; esto supone ensayo y error, dado que la única indicación que se ofrece son las dimensiones relativas de las diversas naves: la posición de disparo adecuada se produce al adquirir su nave aproximadamente el mismo tamaño que las naves enemigas. Durante esta fase del juego, ¡un movimiento rápido del dedo disparador es una clara ventaja!

Al retirarse otra vez del espacio, su nave regresa a la fortaleza espacial. Esta vez el agujero en la pared es más pequeño y hacer blanco es más difícil. Entonces se encontrará frente a frente con el robot Zaxxon, que avanza hacia usted a través de una rejilla. En este momento, su nave se detiene por completo y su única defensa es vencer al robot antes de que éste lo coja. Ello requiere tres blancos directos sobre los lanzadores de misiles del autómata. Mientras intenta denodadamente conseguirlos, el robot Zaxxon avanza de forma inexorable: si éste consigue alcanzarlo, usted habrá de regresar al comienzo del juego.

Zaxxon: Para el Spectrum y el Commodore 64
Editado por: US Gold, Unit 10, Parkway Industrial Estate, Hineage St, Birmingham B7 4LY, Gran Bretaña
Autor: Peter Adams
Palanca de mando: Necesaria
Formato: Cassette (Spectrum, Commodore); disco (sólo Commodore)

Tony Sleep

Tortuga didáctica
A través de este ejercicio con la tortuga Valiant, se está introduciendo a estos alumnos en los principios de la geometría. Esos niños toman un papel activo en la preparación del laberinto y en la programación de la tortuga para que avance evitando los obstáculos. El ejercicio enseña los principios del desplazamiento angular y otros conceptos abstractos que a los niños pequeños con frecuencia les resultan incomprensibles y nada atractivos

Poder para el alumno

Los robots, además de divertir, pueden representar un método de enseñanza nuevo y eficaz

El empleo del robot en el campo de la educación es un intento por abordar el problema (en especial, para los niños pequeños) que implica la comprensión de conceptos abstractos sin poder recurrir a una representación figurativa. Las nociones complejas de trigonometría, por ejemplo, se pueden demostrar físicamente; los procesos mentales se pueden representar mediante la realidad tridimensional de los movimientos de una tortuga. Asimismo, puede ser una útil ayuda para enseñar los principios de la programación de ordenadores.

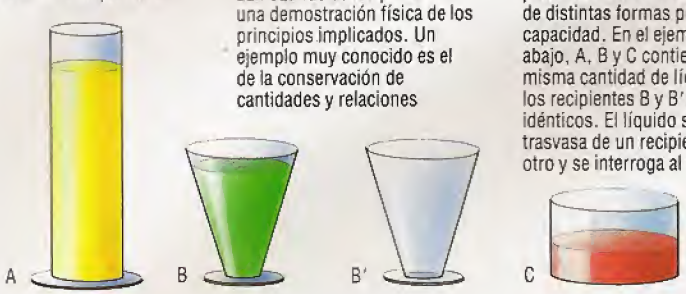
En el pasado, la geometría se ha venido enseñando con instrumentos tales como lápices, reglas y objetos similares. Un niño de ocho años de edad no suele ver mayor sentido en aprender a medir un ángulo, por no hablar de comprender los conceptos más amplios de la geometría. Los ángulos, por otra parte, adquieren muchísimo más significado cuando determinan que la tortuga, programada por el estudiante, choque o no contra el bosque de botellas de plástico colocadas sobre el suelo del aula. Aunque la geometría podría continuar resultando algo nebulosa, al menos hay una motivación para entender sus aplicaciones prácticas. La diferencia entre 15 y 20 cm se vuelve evidente enseguida tras la elección por parte de un alumno entre un ángulo de 5° y otro de 8°.

Uno de los principales aspectos de la tecnología de tortuga es que permite el aprendizaje mediante ensayo y error. El niño puede modificar la programación hasta conseguir trazar un recorrido satisfactorio, adquiriendo mientras tanto una visión constante de las propiedades de ángulos, líneas y mediciones. Un error sólo implica que se debe intentar algo más para producir los movimientos deseados, y no que se ha fallado en algo. Programar un brazo-robot para que levante y mueva un objeto exige pensar en términos de tres planos de movimiento. Imaginar lo mismo sobre el papel, que es bidimensional, no es muy fácil. Con un brazo-robot, sin embargo, los planos de movimiento resultan claramente visibles. Los resultados de la programación se observan a simple vista, los errores se distinguen fácilmente y se pueden efectuar las modificaciones pertinentes hasta conseguir el movimiento correcto. El robot le proporciona al programador una realimentación instantánea, demostrando cada ángulo y medición en secuencia con muchísima más eficacia que un ejercicio equivalente en que se utilizan los tradicionales lápiz y papel.

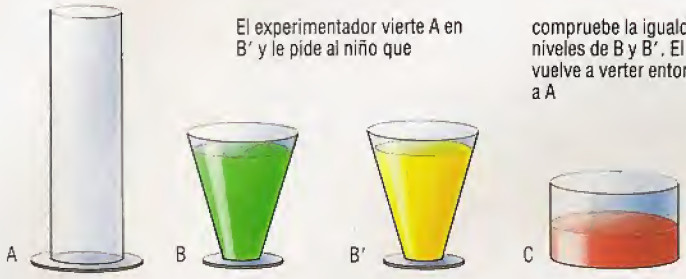
Un juguete robótico que en la actualidad es muy popular en las escuelas (de preescolar a primer grado) es Big Trak, un tanque futurista con una pequeña memoria a bordo y un teclado de calcula-

Ejemplo concreto

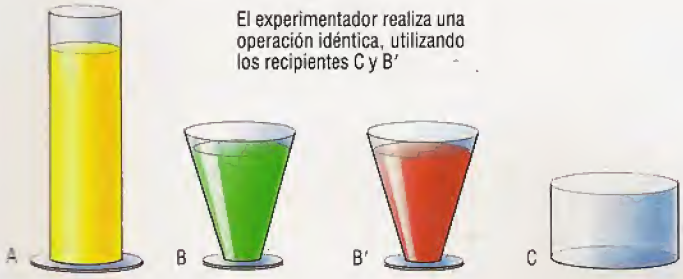
Los niños pequeños a menudo son incapaces de captar conceptos abstractos, aun cuando se les presente una demostración física de los principios implicados. Un ejemplo muy conocido es el de la conservación de cantidades y relaciones transitivas, en los que se le presentan al niño recipientes de distintas formas pero igual capacidad. En el ejemplo de abajo, A, B y C contienen la misma cantidad de líquido, y los recipientes B y B′ son idénticos. El líquido se trasvasa de un recipiente a otro y se interroga al niño



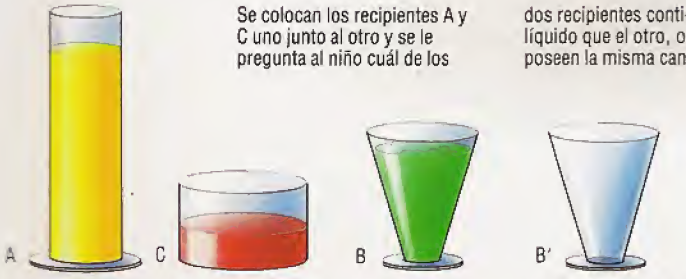
El experimentador vierte A en B′ y le pide al niño que compruebe la igualdad de los niveles de B y B′. El líquido se vuelve a verter entonces de B′ a A



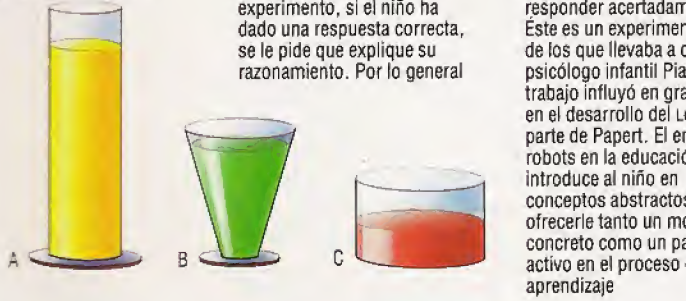
El experimentador realiza una operación idéntica, utilizando los recipientes C y B′



Se colocan los recipientes A y C uno junto al otro y se le pregunta al niño cuál de los dos recipientes contiene más líquido que el otro, o si ambos poseen la misma cantidad



Se colocan sobre la mesa los recipientes A, B y C y se le pregunta al niño si los recipientes contienen la misma cantidad de agua. En las dos últimas etapas del experimento, si el niño ha dado una respuesta correcta, se le pide que explique su razonamiento. Por lo general los niños de menos de siete años son incapaces de captar los principios implicados o de responder acertadamente. Éste es un experimento típico de los que llevaba a cabo el psicólogo infantil Piaget, cuyo trabajo influyó en gran medida en el desarrollo del LOGO por parte de Papert. El empleo de robots en la educación introduce al niño en conceptos abstractos, al ofrecerle tanto un modelo concreto como un papel más activo en el proceso del aprendizaje



Kevin Jones

dora arriba. Teclas con flechas determinan la dirección en que se moverá el aparato, y esta instrucción ha de ir seguida por un número del 1 al 99. Forward 1 impulsará a Big Trak hacia adelante una distancia equivalente a su longitud, y Right 15 hará que efectúe un giro de 15 "minutos", o 45°, hacia la derecha. Se pueden almacenar hasta ocho instrucciones a la vez. Se lo emplea como sustituto de la tortuga, como estímulo para que el niño elabore rutinas simples de programación. El Big Trak demuestra que las instrucciones se llevan a cabo por el orden en el cual se impartieron: una iniciación muy útil a la programación de ordenadores.

Si bien las tortugas pueden ser de ayuda para la enseñanza de las matemáticas, ésta no es su función primordial; fueron diseñadas para enseñar al niño cómo programar un ordenador. El creador de la tortuga, Seymour Papert, cree que los niños deben programar el ordenador en vez de permitir que sea éste el que los programe a ellos. Esta filosofía queda patente en el lenguaje para ordenador de Papert, el LOGO, diseñado para el aprendizaje. Papert define la tortuga como "un objeto con el que pensar", en virtud del cual el niño aprende los principios básicos de la programación mediante el empleo del LOGO para controlar la tortuga.

Paul Cheung, de la Universidad de Edimburgo, desarrolló el aspecto robótico del LOGO, habiendo advertido la necesidad de un software más versátil en una época en que la mayoría de las personas estaban concentradas en el desarrollo de hardware. Cheung quería que los niños pudieran controlar diversos robots. El LOGO es ideal para programar tortugas, pero es limitado para programar otros dispositivos. Tomando el LOGO como punto de partida, Cheung reestructuró el lenguaje permitiéndole controlar más funciones y reforzando su capacidad para recibir entradas desde dispositivos tales como sensores táctiles, lectores de códigos de barras y detectores de luz. El resultado fue el Concurrent-LOGO, o C-LOGO, un lenguaje capaz de "proceso en paralelo" (la capacidad de ejecutar varias funciones simultáneamente).

El C-LOGO puede controlar hasta ocho procesos al mismo tiempo. Dos funciones particularmente interesantes son FOREVER y WHENEVER. La primera, FOREVER, iniciará un proceso y dejará que el mismo continúe ejecutándose hasta que el C-LOGO lo termine. WHENEVER espera a que aparezca una condición, momento en el que activará una función determinada; es esencial para el funcionamiento del proceso en paralelo. El C-LOGO permite activar y desactivar interruptores y también detectará si un interruptor está encendido o apagado. Esta capacidad se utiliza con el sensor: si un sensor táctil entra en contacto con una obstrucción, hace que se "encienda" un interruptor, informando al ordenador de que se ha establecido un contacto. Los niños pueden programar un *buggy* para que evite obstáculos, utilizando la función WHENEVER para ordenarle al robot que gire cada vez que se produzca un contacto.

El C-LOGO es más adecuado para los niños de escuela secundaria, donde Paul Cheung lo utilizó con diversos dispositivos de control. Niños de entre 13 y 15 años de edad han escrito programas para controlar molinos de viento, ascensores, brazos mecánicos y *buggies*, así como un programa para conectar en interface con una caja de botones que controla

los movimientos de puertas y ventanas de una casa robótica. Esta casa contenía una alarma antirrobo que se activaba al abrirse una ventana, y las puertas sólo se podían abrir tras digitar en el ordenador una contraseña. Estos dos programas se ejecutan simultáneamente mediante la incorporación del proceso en paralelo.

A las tres plantas de la casa se accedía mediante el ascensor, que estaba programado para detenerse en cada planta. No obstante, el ordenador debía saber en qué planta se hallaba el ascensor, con el fin de que el mismo funcionara correctamente. Construido con un juego de construcción Meccano, el ascensor estaba alimentado por un motor CD que tenía instalados tres interruptores de lengüeta que detectaban si el ascensor se hallaba en una planta determinada. Los niños lo programaron luego para que fuera a la planta baja pulsando un botón (de la caja de botones), otro para la segunda planta, y un tercero para la planta superior. Los niños se plantearon esta pregunta: "¿Qué sucederá si se pulsa el botón estando el ascensor en movimiento?" Su respuesta inmediata fue indicarle que no hiciera caso a ninguna señal hasta no estar detenido. Pero después de pensarlo un poco más, los niños decidieron que sería una mejor solución hacer que el ordenador llevara el registro de las instrucciones y las implementara en secuencia.

Soluciones simples

Fueron las características del C-LOGO las que les permitieron a los niños escribir de forma muy simple un programa que, de lo contrario, habría sido complejo. Se definieron como componentes esenciales un detector de señales, un secuenciador (*scheduler*) y un controlador de ascensor, actuando el detector de señales y el detector del ascensor como procesos paralelos. Las señales se detectaban utilizando FOREVER y se le pasaban luego al secuenciador. El controlador del ascensor le enviaba entonces un mensaje al secuenciador pidiéndole un destino, con lo que el ascensor se trasladaba a la planta apropiada y después solicitaba otro destino. Tras almacenar las instrucciones en secuencia, el secuenciador las enviaba por orden al controlador. Entusiasmados por tener el ascensor a sus órdenes, así como por contar con los medios para controlar su programación, los niños perseveraron hasta resolver el problema.

Muchos especialistas en educación piensan que uno de los beneficios que le reporta al niño la programación de robots es que ciertos "errores" se pueden eliminar de sus procesos mentales. La experiencia del ensayo y error, y el consiguiente desarrollo de un método de planificación más secuencial, hace que la condición del aprendizaje pase de una mentalidad de "bien o mal" al entorno más realista que requiere correcciones y modificaciones constantes para adecuarse a la situación dada. Los maestros que hoy trabajan con robots ven en ellos el mismo potencial que a comienzos de los ochenta vieron en los ordenadores los maestros que comenzaron a incorporarlos. La aparición del LOGO y el C-LOGO, la caída de precios del hardware y el creciente interés comercial para la robótica por parte de las más importantes firmas informáticas le auguran un brillante futuro al empleo de los robots como medio auxiliar para la educación.

Jugando con la tortuga

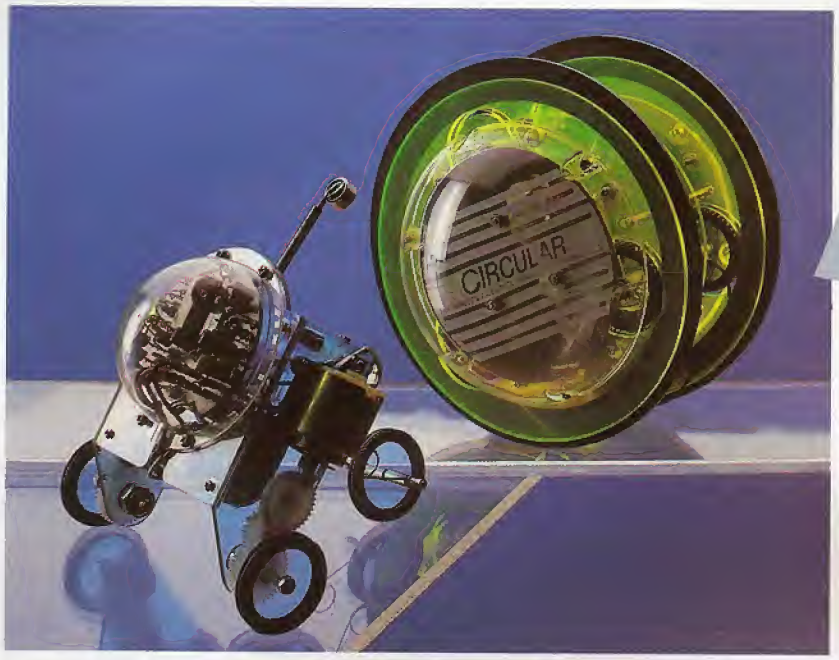
El papel de la tortuga en la clase no se limita a la geometría, la trigonometría y la programación. Se han construido laberintos como parte de un proyecto sobre mitología (Teseo y el Minotauro) y sobre arquitectura religiosa (el laberinto de la catedral de Reims). Un proyecto particularmente ambicioso supuso la construcción de una ciudad a escala en las clases de arte y artesanía, que la tortuga debía recorrer, visitando varias tiendas diferentes para "adquirir" productos incluidos en su "lista de la compra".

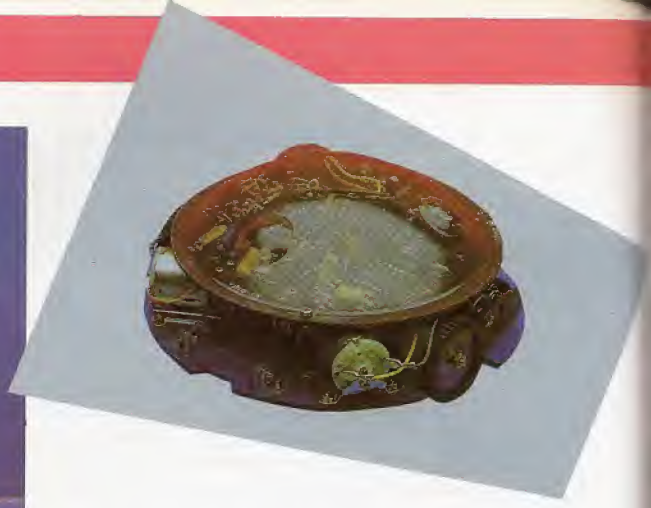
Otros juegos con la tortuga son *Racetrack*, en el que cada niño debe competir con los otros en programarla para que complete el recorrido de una pista de carreras sencilla, y *Postman*, en el que los niños se sientan en círculo y se envían entre sí mensajes escritos, que reparte la tortuga.

Turnturtle y *Shoveturtle* son dos juegos diseñados para ayudar a enseñar el desplazamiento angular y la distancia. En el primero, la tortuga se coloca en el centro de un círculo dividido en secciones. Cada sección se señala con un número diferente (marcador) y el niño ha de programar la tortuga para que gire y se detenga señalando hacia una de las secciones. Gana quien obtenga el marcador más alto con la menor cantidad de instrucciones. *Shoveturtle* utiliza un método similar pero siguiendo una pista recta.

Tanque programable
El Big Trak es una unidad autocontenida que se puede programar al estilo LOGO utilizando su teclado numérico. Las instrucciones entradas se componen de una dirección (izquierda, derecha, etc.) seguida por la cantidad de pasos (hasta 99). Big Trak puede recordar y efectuar de forma secuencial hasta ocho instrucciones diferentes

Ian McKinnell





Trío futurista
Aquí tenemos los tres "robots" más caros de la gama Movits. Son, de izquierda a derecha, el Piper Mouse, el Circular y el Memocon Crawler. Cada máquina de la gama se halla dentro de una atractiva carcasa de plástico transparente, que deja a la vista los mecanismos internos (los motores y los componentes electrónicos que activan los dispositivos) y le confiere a la máquina un adecuado aire "futurista"

Nombre: ¿robot?

Los Movits son "kits" de robot, económicos y preprogramados, de gran demanda en el floreciente mercado de este tipo de productos

La predicción general apunta a que la robótica será el próximo paso en la evolución hacia el hogar informatizado. La etapa de desarrollo que ha alcanzado en la actualidad la robótica se está comparando a la del microordenador a comienzos de los años setenta, con las tecnologías que van surgiendo combinándose entre sí para formar un producto enteramente nuevo. Mientras que el microordenador se desarrolló en una relativa oscuridad, con apenas un puñado de aficionados a la electrónica comprendiendo el potencial de las nuevas máquinas, en la actualidad son muchas más las personas conscientes de la actual revolución de la microelectrónica.

Dado el amplio interés que existe por esta tecnología, era sólo cuestión de tiempo que algún fabricante emprendedor apareciera con una serie de "robots" de costo reducido para el hogar. Los Movits, fabricados por la empresa japonesa Kaho Musen, figuran entre los primeros intentos por satisfacer y rentabilizar el interés del público por la robótica.

Cada Movit viene en forma de modelo para armar (*kit*), sin que sea necesario ningún componente extra, aparte de las pilas de 1,5 V, e incluye un destornillador para facilitar su montaje. Rodeadas por una carcasa de plástico sólido con una cúpula de plástico transparente que deja a la vista sus sistemas de circuitos, las máquinas poseen un sugerente aspecto futurista. En la actualidad hay cinco Movits a la venta, siendo el más barato de todos el Monkey (mono), controlado por sonido. Al activarse a través del sensor de sonido ("escuchando" un grito o una palmada, p. ej.), dos brazos de movimiento alterno empujan el aparato a lo largo de una cuerda o algún otro objeto estrecho. El sensor es un transistor de micrófono condensador conectado mediante una placa de circuito impreso, que decodifica la señal. Las señales apropiadas harán arrancar el motor eléctrico CD, que hace girar un mecanismo de manivela y provoca el movimiento de los brazos. Un reloj situado en la placa asegura que el Monkey se detenga.

El segundo Movit en la escala de precios es el Line Tracer II. Se controla mediante un sensor infrarrojo fijado a su base, y sigue cualquier línea de gran contraste que se dibuje en el suelo. Este robot es impulsado mediante dos motores CD que generan la potencia para sus tres ruedas. A continuación viene el Pier Mouse, también controlado mediante un sensor de sonido, gobernado esta vez por un "silbato para perros". Siguiendo cada soplido del silbato, el Movit ejecuta una instrucción de una secuencia de instrucciones preestablecidas: girar izquierda, parar, girar derecha, parar, avanzar, parar. Siguiendo esta secuencia, el Movit se desplaza en línea recta.

El Circular Movit es la primera de las máquinas con movimiento gobernado completamente a control remoto. Activado por radio, lo impulsan dos motores CD que activan las grandes ruedas de la parte exterior de las burbujas. En la caja que contiene los mandos hay dos botones: cada uno de ellos gobierna uno de los motores. Por consiguiente, para hacer girar el Movit se han de pulsar bien el botón izquierdo, bien el derecho, y ambos botones a la vez para que se desplace hacia adelante. Para el control independiente hay dos placas PCB, cada una de las cuales toma una señal diferente de la caja de control.

Los dispositivos que hemos descrito hasta ahora son atractivos e interesantes de construir, pero la tecnología que utilizan apenas puede describirse como avanzada. El más "vanguardista" de estos robots es el Circular, pero incluso esta tecnología ya está siendo utilizada desde hace algún tiempo, como, por citar algunos ejemplos bien visibles, en los modelos a escala de barcos y aviones controlados por radio que se pueden ver en los parques de las ciudades de todo el mundo. La cualidad esencial

de lo que entendemos por robot es que lleve incorporado algún sistema de guía que se pueda programar para llevar a cabo una secuencia de acciones.

El Memocon Crawler es una máquina que se acerca mucho a nuestra definición de lo que es un robot y, al igual que otros Movits, está alimentado por dos motores eléctricos CD. El sistema de guía se compone de un teclado conectado al Crawler mediante un cable plano. El teclado posee cinco interruptores separados por cada una de las cinco instrucciones disponibles: adelante, izquierda, derecha, pausa y hacer sonar un zumbador e iluminar los LED de la máquina. Estas instrucciones se almacenan en un chip de RAM estática con una memoria que contiene 256 bytes, constando cada byte de cuatro bits. Los bytes que retienen las instrucciones no son direccionales de forma individual y se debe acceder a ellos en secuencia. Esto significa que no se pueden repetir pasos dentro de la secuencia, aunque, por supuesto, se puede volver a ejecutar toda la secuencia entera otra vez.

Dado que vienen en forma de modelo para armar y es necesario montarlos, a muchas personas no les interesará comprarse un Movit. Sin embargo, a pesar de la pasmosa cantidad de componentes que se proporcionan con cada *kit*, en realidad la construcción de las máquinas es notablemente sencilla, e incluso los niños pueden montarla. De hecho, un alto ejecutivo de una de las principales empresas distribuidoras hizo hincapié en que su hijo, de diez años de edad, era el principal montador de los modelos de demostración de la empresa.

Los diversos componentes de los robots vienen en bolsas separadas que llevan marcado un número. Las instrucciones consisten en un esquema del Movit. Cada pieza está dibujada por separado e identificada claramente por el número de su bolsa; mediante flechas se señala con exactitud dónde va cada componente. A la mayoría de las personas esta ayuda les resulta suficiente para completar el robot; pero hay, además, instrucciones escritas que le serán de ayuda en cualquier etapa de la construcción que le resulte confusa. Puesto que la placa de circuito impreso ya está montada, las únicas herramientas que se requieren son un par de llaves, un destornillador y un pequeño martillo.

Construir los Movits es, qué duda cabe, divertido, y los dueños obtendrán muchísima satisfacción al ver moviéndose por el suelo un robot que ellos mismos han ensamblado. Lo que ya es más dudoso es si los Movits representan realmente una entrada en el mundo de la robótica. No obstante, son económicos y, aunque no enseñen gran cosa sobre robótica, podrían proporcionarles a algunas personas, en especial a los niños, una iniciación en este tema, tan interesante y cada vez más popular.

Rodeado por círculos
El Circular, al igual que los otros integrantes de la gama, es fácil de construir. Las placas de circuito impreso para el robot propiamente dicho y la caja de control remoto ya están montadas. Esto significa que el proceso de construcción no exige soldaduras y que quien desee montar su propio Movit no necesitará conocimientos de electrónica para llevar a cabo su propósito

Chris Stevens

MONKEY

MOVIMIENTO
Dos brazos prensores que se mueven alternativamente, activados por movimiento de manivela

CONTROL
Sensor de sonido

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V

LINE TRACER II

MOVIMIENTO
Tres ruedas activadas por dos motores CD

CONTROL
Sensor infrarrojo

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V y una pila de 9 V

PIPER MOUSE

MOVIMIENTO
Tres ruedas activadas por dos motores CD

CONTROL
Sensor de sonido

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V y una pila de 9 V

CIRCULAR

MOVIMIENTO
Dos ruedas con engranaje de dientes activadas por dos motores CD

CONTROL
Caja de control por radio

FUENTE DE ALIMENTACION
Tres pilas de 1,5 V, una pila de 9 V y otra pila de 9 V para la caja de control

MEMOCON CRAWLER

MOVIMIENTO
Tres ruedas activadas por dos motores CD

CONTROL
Memoria programable

FUENTE DE ALIMENTACION
Dos pilas de 1,5 V y una pila de 9 V

Lista de mercancías

El Nuevo Mundo
La era de los grandes descubrimientos se inicia a fines del s. XV. En el s. XVI, los exploradores europeos estaban apenas empezando a elaborar los mapas de los continentes. Al principio estas cartas de navegación eran esquemáticas y a menudo las islas (e incluso continentes enteros) se colocaban en lugares equivocados. A medida que vaya avanzando nuestra simulación, iremos revelando una parte diferente del mapa del mundo, tal como los primeros exploradores fueron uniendo entre sí las piezas de las masas de tierra del globo

Ahora necesitamos adquirir algunos artículos más, indispensables para el éxito de la expedición

Los salarios de la tripulación se abonarán sólo si el barco regresa a puerto, y en ese momento. El jugador debe decidir cuánto invertir en mercancías para comerciar, con la esperanza de obtener un beneficio, y cuánto oro dejar en reserva (o no) para ayudar a pagar los salarios de la tripulación cuando se regrese. Las mercancías disponibles son medicinas, armas, sal, tela, cuchillos y joyas; y se informa al jugador sobre sus precios y se le pregunta la cantidad a comprar. Al cabo de cada transacción se imprime el balance de dinero. Si no hay suficientes fondos para pagar el pedido, así se le informará al jugador y se le solicitará que vuelva a probar. La bodega del barco tiene capacidad ilimitada.

Al igual que en las dos etapas de preparación previas, las matrices para la tercera etapa, que representan los nombres, precios y cantidades de las mercancías compradas, se DIMensionan al comienzo del programa. Las mismas se añaden a las sentencias de inicialización anteriores, comenzando en la línea 30. La cantidad de cada tipo adquirido se retendrá en la matriz OA(). Las descripciones de las otras mercancías están retenidas en la matriz D$() DIMensionada en la línea 31. Los elementos de esta matriz se establecen en las dos líneas siguientes. El primero, D$(1), se establece como FRASCO DE MEDICINA. Los cinco elementos restantes, de D$(2) a D$(6), se establecen en ARMA, BOLSA DE SAL, BALA DE TELA, CUCHILLO y JOYA. Como antes, la referencia a un objeto por su dirección en la matriz facilitará la subsiguiente programación.

El costo de estas mercancías se DIMensiona en una matriz, OC (), que contiene seis elementos, en las líneas 34 y 35. El primer elemento, OC(1), representa el precio de las medicinas, que valen una pieza de oro el frasco. Las armas valen cinco piezas de oro (OC(2)=5), y con una pieza de oro se pueden comprar cinco bolsas de sal (OC(3)=0.2). Una bala de tela cuesta 2 piezas de oro (OC(4)=2), los cuchillos, media pieza de oro (OC(5)=0.5), y las joyas, una pieza de oro cada una (OC(6)=1).

En el programa principal, la línea 500 llama a la rutina para contratar la tripulación, y la línea 550 llama a la subrutina de aprovisionamiento. Ahora se debe insertar la línea 600 para llamar a la subrutina que se encarga de la adquisición de las otras mercancías. Veamos de forma detallada esta subrutina (líneas 3000-3999). Tras limpiar la pantalla hay una demora, y las líneas 3005 y 3010 imprimen y subrayan el título. Se le dice entonces al jugador que para el viaje se necesitarán otras mercancías.

La programación para la compra de los seis tipos de mercancías está organizada en un bucle. El bucle se utiliza una vez para cada tipo, empleando T como el contador de bucle de 1 a 6. Se le dice al jugador el costo de cada producto y su nombre se

representa mediante D$(T) en la línea 3070. La primera vez que se realice el bucle, T se establecerá en 1, para el primer producto: medicinas. El costo por unidad del producto comercial actual se establece mediante OC(T) en la línea 3080. Cuando T se establece en 1, OC(T) representará una pieza de oro, que es el costo de un frasco de medicina. La segunda vez que se ejecute el bucle, T se habrá incrementado a 2, y D$(T) se establecerá en ARMA y el costo, OC(T), será cinco piezas de oro, y así sucesivamente hasta que T alcance el valor 6.

En función de si el costo es de una o más piezas de oro, se imprime el mensaje en singular o en plural, en las líneas 3085 y 3086. A continuación se produce una demora y se le pregunta al jugador: "QUIERES COMPRAR? (S/N)". La línea 3110 espera una respuesta, P$. La línea 3120 comprueba el carácter situado más a la izquierda de la respuesta. Si es "N" se lo envía a la línea 3175, enviando el programa hacia atrás hasta el comienzo del bucle, para presentarle el siguiente producto. Si la entrada es "S" o "SI", hay una demora y la línea 3130 pregunta "CUANTO QUIERES?". La línea 3135 espera una respuesta y el jugador puede responder digitando 10, o 10 BOLSAS DE SAL. Esta elección es posible porque la línea 3140 examina la respuesta y emplea VAL para reconocer los caracteres numéricos al comienzo de la frase e ignorar el texto.

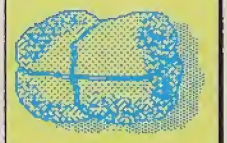
Si usted trata de hacer una compra ilegal, sin tener oro suficiente para pagar la factura, la línea 45 hace una comprobación para determinar si el costo del producto, OC(T), multiplicado por el número entrado, TT, es mayor que el dinero disponible, MO. De ser así, se envía el programa a la línea 3150, que le informa al jugador acerca de su insolvencia monetaria. Tras una demora, se le dice al jugador "POR FAVOR, VUELVE A ENTRAR", en la línea 3154. El programa retorna a la línea 3130, que pregunta "CUANTO QUIERES COMPRAR?" si es que está en condiciones de abonar el nuevo pedido. Si hay fondos suficientes, el programa va a la línea 3160, donde la cantidad gastada se resta del fondo, mediante la fórmula: MO=MO – (OC(T)*TT). MO representa el dinero restante. El costo del producto, OC(T), se multiplica por la cantidad comprada, TT, y se resta del balance. TT se almacena en la matriz OA() en la línea 3165 y, tras una demora, la línea 3176 imprime una línea en blanco y el balance de oro. Tras otra demora, T se incrementa en 1 y el programa retorna al comienzo del bucle mediante la línea 3200, para ofrecer el siguiente producto.

Una vez seleccionados todos los productos (y al cabo de una pausa) la línea 3210 le dice al jugador que la etapa de compra ha terminado. La línea 3230 imprime "PULSA CUALQUIER TECLA PARA CONTINUAR" utilizando la variable establecida al comienzo del programa, K$, y aguarda una respuesta antes de retornar al programa principal. La tarea final consiste en resumir las mercancías pedidas y la tripulación contratada antes de pasar al inicio del viaje. Esto lo hace la rutina Resumen de Apertura.

El programa imprime "CUENTAS CON LA SIGUIENTE TRIPULACION" y se imprimen en un bucle las categorías de tripulante, en la línea 645 (FOR T=1 TO 5) para cada una de las cinco categorías de la tripulación. La línea 650 comprueba si se ha contratado alguno de esta categoría, comprobando CC(T), el contador de la tripulación para ese tipo. Si el contador es 0, la línea 670 envía el programa hacia atrás, hasta el comienzo del bucle, incrementando en uno el valor de T. Si se ha contratado a algún tripulante de esa categoría, se imprime la cantidad y la categoría. Tras la descripción se imprimen una S o un espacio, realizando el programa cinco veces el bucle, una para cada categoría de tripulante. De modo similar se imprime una lista de las provisiones, entre las líneas 675 y 710, utilizando un bucle de 4 (línea 685) para las cuatro clases de provisiones.

El resto de las mercancías adquiridas no se imprime en un bucle porque sus descripciones son muy diferentes. Éstas no se describen por unidades, por lo cual la simple adición de una "S" no creará necesariamente una palabra en plural; una bala de tela se convierte en balas de tela, por ejemplo. Cada una de las mercancías extras se trata por separado de forma lineal. Si no se ha comprado nada de algún tipo de mercancía, el programa pasa directamente a la siguiente categoría. La línea 730, por ejemplo, comprueba si se han comprado medicinas y, si no, el programa salta hasta la sección siguiente, en la línea 740. Si se han comprado medicinas,

Las mercancías esenciales
Éstas son las mercancías que nuestros expedicionarios podrían haber esperado llevar consigo para comerciar con los habitantes de las nuevas tierras. Por supuesto, también habrán tenido que abonar las mercancías antes de hacerse a la mar con ellas. En nuestra simulación, los nombres de cada una de las posibles mercancías transportadas están retenidos en la matriz D$. El precio por unidad está retenido en la matriz OC, mientras que la cantidad adquirida está retenida en la matriz OA

	(1)	(2)	(3)	(4)	(5)	(6)
D$	 Frasco de medicina	 Arma	 Bolsa de sal	 Bala de tela	 Cuchillo	 Joya
OC	1	5	1/5	1 1	1/2	1
OA	6	20	15	32	20	10

el programa determina si fueron un frasco o más, e imprime la terminación adecuada: una S o un espacio. Luego imprime la cantidad y pasa a la siguiente categoría. Una vez completada la lista, la línea 792 informa al jugador sobre la cantidad de dinero que le queda. Se le dice al jugador "PULSA CUALQUIER TECLA PARA COMENZAR EL VIAJE" en la línea 797, y entonces el programa espera alguna entrada.

Módulo Tres: Mercancías para comerciar

Dimensionamiento de las matrices

```
30 DIM OA(6)
31 DIM D$(6)
32 D$(1)="FRASCO DE MEDICINA":D$(2)="ARMA":D$(3)=
   "BOLSA DE SAL"
33 D$(4)="BALA DE TELA":D$(5)="CUCHILLO":D$(6)=
   "JOYA"
34 DIM OC(6)
35 OC(1)=1:OC(2)=5:OC(3)=.2
36 OC(4)=2:OC(5)=.5:OC(6)=1
```

Llamada a Subrutina Mercancías para Comerciar

```
600 GOSUB 3000
```

Rutina Resumen de Apertura

```
605 REM **** LISTO PARA EMPEZAR ****
610 PRINTCHR$(147)
615 S$="AHORA ESTAS LISTO PARA EMPEZAR*":GOSUB 9100
625 S$="EL VIAJE.*":GOSUB 9100
630 GOSUB 9200
635 PRINT:S$="CUENTAS CON LA SIGUIENTE
    TRIPULACION:*":GOSUB 9100
640 GOSUB 9200
645 FOR T=1 TO 5
650 IF CC(T)=0THEN 670
655 PRINT CC(T);
660 PRINT C$(T);
662 IF CC(T)=1THEN PRINT" ":GOTO 668
664 PRINT "S"
668 GOSUB 9200
670 NEXT
674 GOSUB 9200
675 PRINT:S$="Y LAS SIGUIENTES PROVISIONES:*":GOSUB
    9100
680 GOSUB 9200
685 FOR T=1 TO 4
690 IF PA(T)=0 THEN 710
695 PRINT PA(T);U$(T);"S DE";
700 PRINT P$(T)
708 GOSUB 9200
710 NEXT
715 GOSUB 9200
720 PRINT:S$="TAMBIEN POSEES:*":GOSUB 9100
725 GOSUB 9200
730 IF OA(1)=0 THEN 740
733 IF OA(1)=1 THEN S$="FRASCO DE MEDICINA*":GOTO 735
734 S$="FRASCOS DE MEDICINA*"
735 PRINT OA(1);:GOSUB 9100
736 GOSUB 9200
740 IF OA(2)=0THEN 750
743 IF OA(2)=1 THEN S$="ARMA*":GOTO 745
744 S$="ARMAS*"
745 PRINT OA(2);:GOSUB 9100
746 GOSUB 9200
750 IF OA(3)=0 THEN 760
753 IF OA(3)=1 THEN S$="BOLSA DE SAL*":GOTO 755
754 S$="BOLSAS DE SAL*"
755 PRINT OA(3);:GOSUB 9100
756 GOSUB 9200
760 IF OA(4)=0 THEN 770
763 IF OA(4)=1 THEN S$="BALA DE TELA*":GOTO 765
764 S$="BALAS DE TELA*"
765 PRINT OA(4);:GOSUB 9100
766 GOSUB 9200
770 IF OA(5)=0 THEN 780
773 IF OA(5)=1 THEN S$="CUCHILLO*":GOTO 775
774 S$="CUCHILLOS*"
775 PRINT OA(5);:GOSUB 9100
776 GOSUB 9200
780 IF OA(6)=0 THEN 790
783 IF OA(6)=1 THEN S$="JOYA*":GOTO 785
784 S$="JOYAS*"
785 PRINT OA(6);:GOSUB 9100
786 GOSUB 9200
790 GOSUB 9200
792 PRINT:PRINT"TE QUEDAN ";MO;"PIEZAS DE ORO"
796 GOSUB 9200
797 S$="PULSA CUALQUIER TECLA PARA COMENZAR EL
    VIAJE*"
798 GOSUB 9100
799 GET P$:IF P$=""THEN 799
999 END
```

Subrutina Mercancías para Comerciar

```
3000 REM **** ETAPA 3 OTRAS MERCANCIAS ****
3001 PRINTCHR$(147):REM ETAPA 3
3002 GOSUB 9200
3005 PRINT"     ETAPA 3 - OTRAS MERCANCIAS"
3010 PRINT"     ----- - - ----- ----------"
3020 GOSUB 9200
3025 PRINT
3030 S$="HAY OTRAS COSAS QUE PODRIAN*":
     GOSUB 9100
3035 S$="SERIE UTILES PARA EL VIAJE,POR*":
     GOSUB 9100
3040 S$="EJEMPLO MEDICINAS Y MERCANCIAS*":
     GOSUB 9100
3045 S$="PARA COMERCIAR.*":GOSUB 9100
3046 GOSUB 9200
3050 S$="PUEDES NECESITAR TAMBIEN ESCOPETAS.*":
     GOSUB 9100
3055 GOSUB 9200:GOSUB 9200
3060 FOR T=1 TO 6
3065 PRINT
3070 PRINT"UN";D$(T);
3075 S$="CUESTA*":GOSUB 9100
3080 PRINT OC(T);
3085 IF OC(T)=1 THEN PRINT "PIEZA DE ORO":GOTO
     3090
3086 PRINT "PIEZAS DE ORO"
3090 GOSUB 9200
3095 S$="TE GUSTARIA COMPRAR (S/N)*":GOSUB 9100
3110 INPUT P$:P$=LEFT$(P$,1)
3115 IF P$<>"S" AND P$<>"N" THEN 3095
3120 IF P$="N" THEN 3175
3125 GOSUB 9200
3130 S$="CUANTO QUIERES*":GOSUB 9100
3135 INPUT P$
3140 TT=VAL(P$)
3145 IF OC(T)*TT>MO THEN 3150
3147 GOTO 3160
3150 S$="NO TIENES DINERO SUFICIENTE*":GOSUB 9100
3152 GOSUB 9200
3154 S$="POR FAVOR VUELVE A ENTRAR*":GOSUB 9100
3155 GOSUB 9200:GOTO 3130
3160 MO=MO - (OC(T)*TT)
3165 OA(T)=TT
3170 GOSUB 9200
3175 PRINT
3176 PRINT" DINERO SOBRANTE = ";MO
3200 GOSUB 9200:NEXT T
3205 GOSUB 9200:PRINT:PRINT
3210 S$="FIN DE LA ETAPA 3*":GOSUB 9100
3220 GOSUB 9200:PRINT
3230 S$=K$:GOSUB 9100
3240 GET P$:IF P$=""THEN 3240
```

Complementos al BASIC

Spectrum:
Realice las siguientes modificaciones:

```
  32 DIM D$(6,20)
 610 CLS
 799 LET P$=INKEY$:IF P$="" THEN GO TO 799
3001 CLS
3110 INPUT P$:LET P$=P$(1 TO 1)
3240 LET P$=INKEY$: IF P$="" THEN GO TO 3240
```

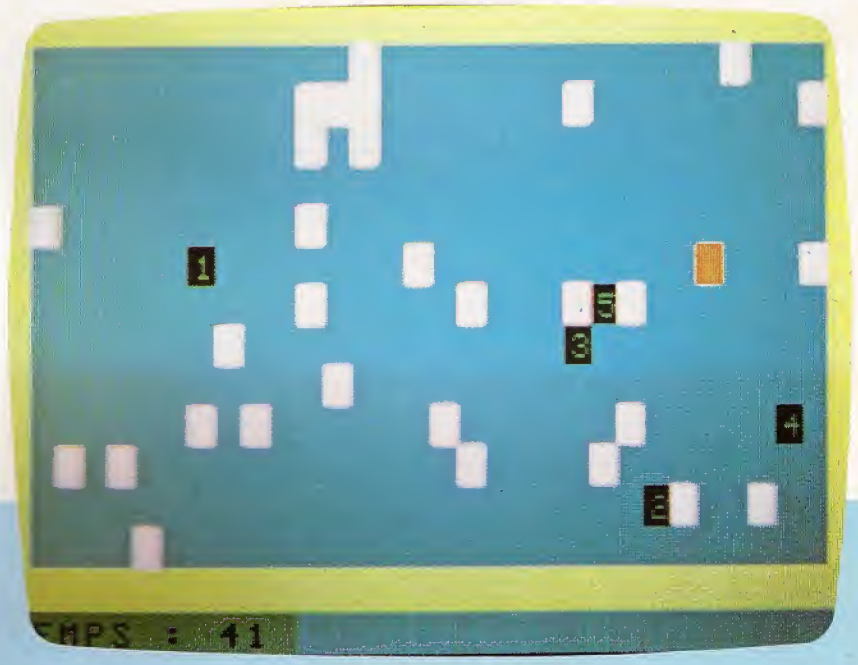
BBC Micro:
Realice las siguientes modificaciones:

```
 610 CLS
 799 A$=GET$
3001 CLS
3240 A$=GET$
```

Cifras

He aquí un juego de destreza más difícil de lo que parece. Esta versión es para el microordenador Alice, de Matras

Usted debe tratar de obtener la mayor cantidad de puntos posible al borrar, con la ayuda del cuadrado rojo, las cifras que muestra el ordenador. Las teclas a utilizar son W (abajo), Q (izquierda) y S (derecha). La barra espaciadora le permite detenerse. *Atención:* Al borrar los números debe seguir un orden (de menor a mayor), evitando los obstáculos colocados al azar en la pantalla, y hacerlo en un período de tiempo limitado. (El tiempo que le va quedando se visualiza en la última línea.) Cuando todas las cifras hayan sido borradas, el juego se reanuda con un número más. Al pasar de las nueve cifras se añade una dificultad: debe borrar los símbolos acordándose del orden en que han aparecido (¡a no ser que no se sepa de memoria el código ASCII!).




```
  5 REM *************
 10 REM *    CIFRAS   *
 15 REM *************
 20 S=0
 30 X=0
 40 P$=CHR$(191)
 50 GOSUB 2000
100 $FOR I=1 TO X
110 D$=INKEY$
120 D(D$="Q")-(D$="S")+32*((D$=
   "Z")-(D$="W"))
130 IF D<>0 THEN D0=D
140 IF D$=" " THEN D0=0
150 T=T-0.1
160 PRINT@ 480, "TIEMPO :";INT(T+1);
170 IF T<0 THEN 500
180 P=P+D0
190 C=PEEK(16384+P)
200 IF C=I+48 THEN S=S+I:SOUND 1,1: GOTO 260
210 IF C<>223 THEN P=P1
220 PRINT@ P1,CHR$(223);
230 PRINT@ P,P$;
240 P1=P
250 GOTO 110
260 PRINT@ P1,CHR$(223);
270 PRINT@ P,P$;
280 P1=P
290 NEXT I
300 GOSUB 2000
310 GOTO 100
500 IF R<S THEN R=S
510 PRINT@ 166,"TIEMPO TRANSCURRIDO";
520 PRINT@ 234, "PUNTOS :";S;
530 PRINT@ 266, "RECORD :";R;
540 PRINT@ 326, "OTRA ?";
550 D$=INKEY$
560 IF D$="" THEN 550
570 IF D$<>"N" THEN 20
580 END
2000 CLS 6
2010 X=X+1
2020 FOR I=0 TO 31
2030 PRINT@ I,CHR$(159);
2040 PRINT@ 448+I,CHR$(159);
2050 NEXT I
2060 FOR I=1 TO 13
2070 PRINT@ I*32,CHR$(159);
2080 PRINT@ I*32+31,CHR$(159);
2090 NEXT I
2100 FOR I=1 TO 30
2110 GOSUB 3000
2130 PRINT@ P,CHR$(239);
2140 NEXT I
2150 FOR I=1 TO X
2160 GOSUB 3000
2180 POKE 16384+P, I+48
2190 NEXT I
2200 GOSUB 3000
2210 PRINT@ P,P$;
2220 P1=P
2230 T=50
2240 D0=0
2250 RETURN
3000 P=RND(414)+32
3010 IF PEEK(16384+P)<>223 THEN 3000
3020 RETURN
```

Amplios horizontes

El sintetizador Music 500 incorpora una tecnología que hasta ahora no se hallaba al alcance de los micros personales

Chris Stevens

Música de calidad
El paquete Music 500 de Hybrid es, ciertamente, uno de los más avanzados dispositivos de música por ordenador que se han creado para micros personales. Equipado con 16 osciladores y modulación de onda digital, puede competir ventajosamente con sintetizadores mucho más caros

El Music 500 Package, comercializado por Acorn para el BBC Micro, fue diseñado por otra empresa de Cambridge, Hybrid Technology. Se trata de una compleja pieza electrónica alojada en una caja de unidad de disco de color crema. Se conecta al conector para bus de 1 MHz del BBC Micro y a un amplificador estéreo a través de un conector DIN estándar de 5 patillas.

En el interior de la caja hay 16 osciladores, pudiendo el ordenador programar la frecuencia (*pitch*), volumen, equilibrio y forma de onda de cada uno de ellos. Con el software que lo acompaña, el Music 500 supera en rendimiento a numerosos sintetizadores que existen en la actualidad en el mercado y cuyo precio es varias veces superior.

La mayoría de los sintetizadores producen su escala de sonidos por uno de dos métodos. Los más económicos toman un tono simple (una onda senoidal) y luego la distorsionan y la filtran para producir el efecto requerido. Aunque este método puede producir una gama de sonidos razonable, es aún muy limitado, incluso en las manos de un experto. Algunos de los sintetizadores más caros utilizan varias ondas senoidales que se modulan las unas a las otras. La amplitud de una onda se emplea para controlar la frecuencia de otra. Este método de control es muy superior al método de distorsión simple, pero aun así no puede imitar con exactitud la mayoría de los sonidos producidos naturalmente.

El Music 500, sin embargo, produce sus sonidos por un procedimiento completamente distinto. Las formas de onda creadas por este dispositivo se construyen de forma digital para conformar un patrón y se muestrean a una velocidad de 47 000 veces por segundo. Además de esta capacidad para programar cualquier forma de onda, el Music 500 ofrece técnicas de modulación; no obstante, no se limita a la modulación de frecuencia, y el Music 500 puede combinar canales con modulación de anillo y modulación de fase. También puede utilizar otras formas de onda además de las ondas senoidales simples.

Si no fuera por el software que se proporciona con el Music 500, la simple creación de un tono con este dispositivo sería una tarea difícil. Este software se suministra en cassette, si bien se lo puede transferir fácilmente a disco con el programa que se proporciona. Hybrid Technology tiene planeado lanzar próximamente una versión ampliada del software en ROM. El paquete se suministra con AMPLE, un lenguaje de programación de música que ofrece todo lo necesario para sacar el máximo partido del Music 500. La utilización del lenguaje es, en muchos sentidos, similar al FORTH y al LOGO, tratándose de un lenguaje basado en palabras con las cuales los programadores pueden definir palabras clave y procedimientos. Una composición musical podría tener más o menos el siguiente aspecto:

```
10 .%EL VUELO DEL ABEJORRO
20 .preparacion
30 .ejecucion
```

El vocablo *preparacion* es una palabra definida por el usuario que ya se ha programado del modo siguiente:

```
10 .'preparacion'
20 .[
30 .estsint
40 .estbajo
50 .esttambores
60 .estcimbalo
70 .]
```

Los términos con los que se define preparacion ya han sido programados, a su vez, para definir sonidos diferentes; por ejemplo *esttambores* o *estcimbalo*. Se proporcionan formas de onda preestablecidas para ayudarlo a iniciarse con el Music 500; sin embargo, es muy fácil crear formas de onda nuevas. Las formas de onda se pueden definir ya sea en términos de las proporciones relativas de los primeros 16 armónicos del tono, o bien generar en forma matemática. Este último método le otorga al Music 500 la capacidad de producir "ruido" especificando una forma de onda al azar. La definición para el sonido del címbalo, por ejemplo, podría ser la siguiente:

```
10 .'cimbalo'
20 .[
30 .3 WMOD 0 rand! 128 FOR(RAND? INDEX WG!)
   FOR WGC
40 .1 CHAN
50 .3 WAVE UN RM – 1 POS
60 .2 CHAN
70 .3 WAVE 2000 OFFSET 1 POS
80 .]
```

El equivalente de AMPLE para el bucle FOR...NEXT del BASIC establece cada punto de la forma de onda en un valor al azar. Esta forma de onda se utiliza para dos canales productores de sonido (dos osciladores) y el tono del segundo está ligeramente desplazado respecto al primero y en posición estéreo diferente. El resultado es un sonido ruidoso, estrepitoso... como el de un címbalo.

Como alternativa, se podría definir una forma de onda más melodiosa:

```
10 .'sint'
20 .[
30 .WZERO 181 1 WH! 135 2 WH! 181 3 WH!
          62 4 WH! 76 5 WH! 37 6 WH!
          14 7 WH! 3 8 WH!
40 .WHG WGC 3
50 .]
```

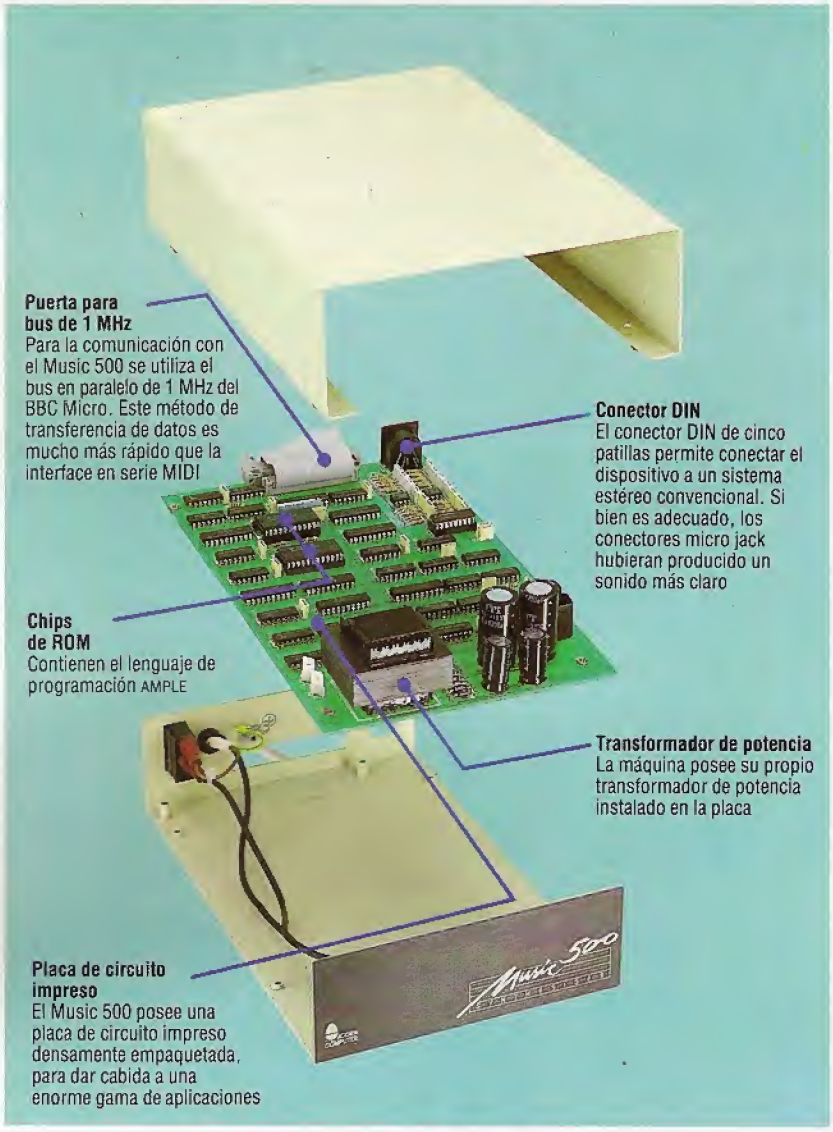
Esta forma de onda de compone de proporciones variables de los primeros ocho armónicos. Las envolturas de tono y amplitud se crean de forma similar, utilizando ya sea las formas simples por defecto o bien las que haya especificado el usuario.

La palabra ejecucion puede llamar a cada uno de los sonidos definidos y demás palabras que contenga la partitura actual.

```
10 .'ejecucion'
20 .[
30 .4 PLAYERS
40 .1 PLAY (melodia para piano)PLAY
50 .2 PLAY (linea de bajo)PLAY
60 .3 PLAY (ritmo de bateria)PLAY
70 .4 PLAY (estruendo de cimbalo)PLAY
80 .GO
90 .]
```

Para explicarlo de forma sencilla, se le da al sistema la cantidad de ejecutantes necesarios, a cada ejecutante se le da un sonido y una partitura (más palabras definidas por el usuario) y por último se les dice que empiecen a tocar, llevando el tiempo entre sí. Esta capacidad del AMPLE para tareas múltiples permite utilizar el Music 500 como cuando se escribe una partitura para orquesta. Uno no necesita instrumentar todas las partes al mismo tiempo, aunque podría hacerlo si lo deseara, y cada parte se trata por separado. Todo el lenguaje es modular y jerárquico. Incluso las palabras incorporadas se pueden volver a definir, aunque el empleo de esta facilidad es bastante complicado. No obstante, se puede, pongamos por caso, redefinir la palabra que marca las barras de modo que la segunda nota de cada barra se toque un poco más fuerte para dar acento.

Si todos estos sonidos son excesivamente técnicos, entonces el AMPLE proporciona algunas palabras de ayuda. En todo momento se ofrecen valores sensatos por defecto, de modo que si sólo desea tocar una melodía, puede limitarse a entrar las notas por el teclado en lugar de dentro de una definición de palabra.



Chris Stevens

Acorn está por sacar un teclado musical completo de 49 notas que se conecta a la puerta para el usuario del BBC Micro, para utilizar con el Music 500. Éste incluirá software en AMPLE, como un "procesador de notas" que grabará lo que usted toque en el teclado y luego le permitirá volver a ejecutarlo y editarlo.

MUSIC 500

Osciladores	16
Gama de frecuencia	0-20 Khz
Resolución de tonos	16avo de semitono
Formas de onda	
Armónicas	16 armónicos
Geométricas	128 puntos
Envolturas (tono/amplitud)	17 segmentos
Longitud de paso	0-320 segundos
Resolución de tiempo	10 ms
Posiciones estéreo	7
Precisión de forma de onda	logarítmica 8 bits
Velocidad de muestreo	46875/seg
Resolución de frecuencia	0,0056 Hz
Base de tiempo	255 ms-655 ms/ciclo
Concurrencia	1-9 procesos

Una y otra vez

El PASCAL cuenta con tres diferentes construcciones de "iteración" para controlar los bucles del programa

Ya hemos analizado dos de los tres métodos del PASCAL para estructuración de sentencias: las estructuras secuenciales que utilizan las "palabras paréntesis" BEGIN y END, y la opción o selección, que emplea las construcciones IF y CASE. El tercer tipo de estructura de sentencias es la *iteración* o repetición, para la que el PASCAL proporciona tres construcciones diferentes. La sentencia FOR da un bucle "activado por contador", mientras que tanto las sentencias WHILE como REPEAT son "activadas por condición".

En primer lugar se evalúa la expresión booleana delimitada por las palabras reservadas WHILE y DO y, si el resultado de la expresión es verdadero, la sentencia siguiente (que puede ser cualquier sentencia en PASCAL, inclusive una sentencia estructurada de cualquier nivel de complejidad) se ejecutará repetidamente mientras la condición permanezca siendo verdadera. Esta condición booleana se vuelve a evaluar tras cada ejecución de la sentencia del "cuerpo" de la construcción WHILE. Obviamente, esto implica que las acciones incluidas en el bucle deben cambiar de alguna forma al menos uno de los elementos evaluados. Veamos un ejemplo:

```
WHILE MaxInt >1 DO
  WriteLn ('Buclando!')
```

tendría enormes dificultades para terminar. Veamos un ejemplo más realista (y más seguro):

```
read (retirar);
WHILE retirar > SaldoBancario DO
  BEGIN
    WriteLn ('Fondos insuficientes — vuelve a
    probar:');
    write ('Cantidad?');
    read (retirar)
  END
```

Este segmento de programa ilustra muy bien una de las propiedades más valiosas de la construcción WHILE. Si, al leerse por primera vez la cantidad a retirar, la misma se halla dentro del saldo actual de la cuenta, el bucle WHILE no se ejecutará en absoluto. Si se ejecuta el cuerpo, la repetición se detendrá cuando la expresión resulte falsa. En este caso, el control del programa seguirá en la primera sentencia tras la construcción WHILE. El aspecto de la expresión booleana del encabezamiento de esta construcción indica sintácticamente la seguridad de la "semántica" (o el significado y comportamiento exactos de las sentencias) del PASCAL. Si usted desea iterar sólo cuando una condición es verdadera, habrá de utilizar el bucle WHILE.

A menudo hay ocasiones en las que necesitamos llevar a cabo el cuerpo de un bucle al menos una vez para preparar la condición que lo detenga, tal como ocurre con el programa Dorada que ofrecimos anteriormente como ejemplo. En estas circunstancias, la construcción REPEAT...UNTIL expresa con más precisión el algoritmo y proporciona una sintaxis natural de alto nivel.

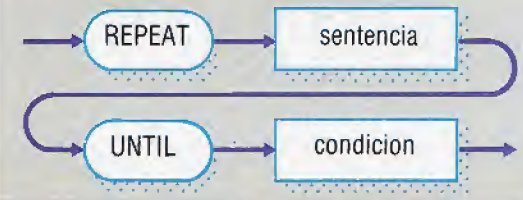
Observe que ahora la condición de terminación aparece al final de la estructura, y que es lógicamente opuesta a la condición utilizada para codificar la construcción WHILE. Es decir, el bucle REPEAT termina cuando la evaluación de la expresión booleana es verdadera, mientras que un bucle WHILE (si es que hubiera comenzado) se detendrá cuando la condición sea falsa. Si el cuerpo de la sentencia REPEAT contiene varias sentencias, éstas, en realidad, deberían encerrarse entre las palabras paréntesis BEGIN...END, pero el PASCAL es elástico en cuanto a esta regla: las palabras reservadas REPEAT y UNTIL serán suficientes para delimitar el cuerpo del bucle. Es perfectamente legal incluir, sin embargo, las palabras superfluas BEGIN y END, algo así como usar un punto y coma extra antes de una palabra reservada, con la excepción de que no se cree ninguna sentencia nula.

El siguiente segmento contará la cantidad de veces que aparezca la letra "e" en una frase entrada desde el teclado. La frase debe terminar con un punto final, momento en el que termina el bucle REPEAT y se imprime el resultado.

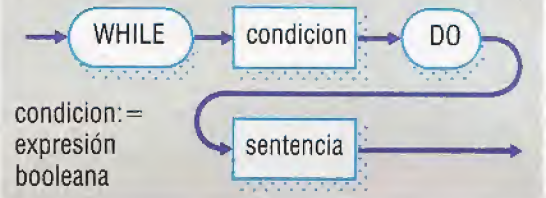
```
contador:=0;
REPEAT
  read (simbolo);
  IF simbolo = 'e' THEN
    contador:=contador+1
UNTIL simbolo='.';
WriteLn ('En la frase hubo', contador: 1, '"e"s.')
```

Aquí no hay preferencia por ninguna construcción y podríamos haber usado un bucle WHILE:

Sentencia REPEAT:



Sentencia WHILE:



Liz Dixon

Respuesta al ejercicio Dorada

En el capítulo anterior planteamos un ejercicio con nuestro programa Dorada, que generaba términos Fibonacci y las proporciones de los pares sucesivos. Le sugerimos que tratara de modificar el bucle de modo que la condición de terminación se detuviera cuando el siguiente término Fibonacci a calcular fuera mayor que MaxInt. Con la representación del complemento a dos, si un resultado entero es mayor que MaxInt, el bit del signo pierde su significado. El número parece entonces negativo, de modo que el programa se podría romper, dado que el tipo Fibonacci excluye los valores negativos. Aun con enteros, el bucle no se detendría jamás. No podemos decir:

UNTIL primero + segundo > MaxInt

porque, por definición, no hay ningún número mayor que MaxInt. No obstante, una reacomodación algo más astuta para efectuar una resta en vez de una suma nos da la solución:

UNTIL segundo > MaxInt — primero

Ahora, por supuesto, ya no se requieren la aritmética real y la constante Epsilon

```
contador:=0;
read (simbolo);
WHILE simbolo <> '.' DO
  BEGIN
    IF simbolo = 'e' THEN
      contador:=succ(contador);
    read (simbolo)
  END;
write ('Hubo', contador : 1,);
WriteLn ('"e"s en la frase.')
```

Se puede ver, sin embargo, la diferencia existente entre las dos estructuras. La acción requerida para determinar la condición de terminación (la sentencia read, en este ejemplo) a menudo se producirá dos veces cuando se utilice una construcción WHILE, una vez inmediatamente antes de entrar a la construcción, y otra vez como la última sentencia del cuerpo del bucle WHILE.

Aunque el bucle WHILE es la única construcción esencial de todos los lenguajes, algunas veces es conveniente tener algún medio de repetir algo una cantidad específica de veces, normalmente a través de cierta gama de valores dentro de una escala. Para crear esta clase de bucles "activados por contador", el PASCAL ofrece una tercera construcción. Aquí el programador de BASIC podría sentirse en un terreno familiar, pero entre la construcción del PASCAL y el bucle FOR del BASIC existen algunas diferencias importantes que hay que tener en cuenta. En primer lugar, como controlador del bucle se puede utilizar cualquier variable escalar y no sólo enteros. En segundo lugar, y lo que es más importante, la sentencia FOR..DO del PASCAL es completamente segura; al igual que el bucle WHILE, puede directamente no ejecutarse (p. ej., si el valor inicial es mayor que el valor final) y existen severas restricciones en el uso de la variable controladora. En especial, es ilegal cambiar este valor en cualquier lugar del cuerpo del bucle. Además, ¡en PASCAL es un error el mero hecho de amenazar con cambiarlo! Cuando veamos procedimientos y funciones podremos apreciar la verdadera ventaja de estas medidas de seguridad; pero, mientras tanto, he aquí un ejemplo ilegal:

```
FOR N:=1 TO 10 DO
  IF N=10 THEN
    N:=1
```

Una soberana tontería, por supuesto. Habiendo solicitado que el bucle se realice diez veces, la sentencia IF volvería a restablecer el valor de N en 1 y crearía un bucle infinito.

Observe que la sintaxis de la construcción FOR..DO tiene una sentencia de asignación entre las dos primeras delimitadoras (FOR y TO) que le asignan el valor inicial a la variable de control del bucle, y el valor final está dado por la expresión encerrada entre las palabras reservadas TO y DO. Estos dos valores deben ser, por supuesto, del mismo tipo escalar que el controlador. Veamos otros ejemplos:

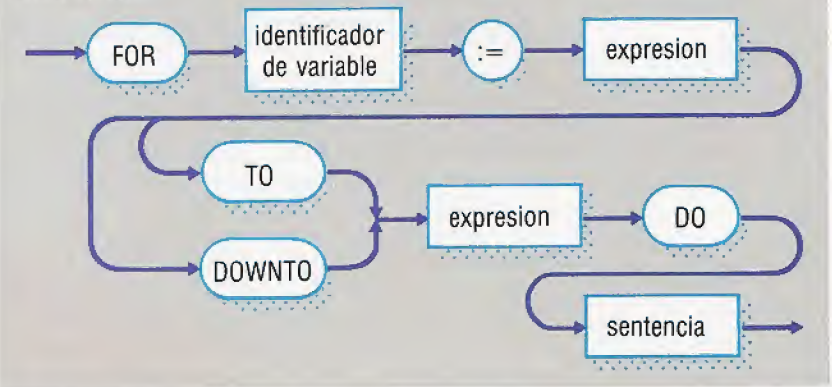
- FOR letra := 'A' TO 'Z' DO {etc}
- FOR mes := Ene TO Dic DO {etc}
- FOR N := N TO succ (MaxInt DIV 1000) DO {etc}

El último ejemplo da por sentado que en algún momento anterior a N ya se le ha asignado un valor; recuerde que si éste era mayor que el valor final especificado, el bucle no se llevaría a cabo. En aquellas ocasiones en las que deseamos descender en una escala de valores, en lugar de TO se utiliza la palabra reservada DOWNTO. De modo, que, por ejemplo, la NASA podría emplear:

```
FOR CuentaAtras :=10 DOWNTO 0 DO
  WriteLn (Cuentaatras : 32 — 3 *CuentaAtras);

WriteLn ('DESPEGUE!')
```

El PASCAL mantiene un control riguroso sobre el bucle FOR y no permite ningún equivalente del incremento y decremento STEP opcional del BASIC.

Sentencia FOR:



Programa Real Ale

```
PROGRAM          RealALe   ( output );

TYPE
  cerveza = (Mild, Bitter, Special, Warmer);
VAR
    precio,
    pintas      : real;
    ale         : cerveza;
BEGIN
  WriteLn ('Pintas' : 6,
      'Mild' : 9, 'Bitter' : 8,
        'Special' : 8, 'Warmer' :8);
  WriteLn;
  pintas :=0.5; {empezar por media pinta}

  REPEAT
    IF pintas - trunc (pintas) >0.4
      THEN {escribir como ##.#}
        write ( pintas : 6 : 1, ' ' )
      ELSE {escribir como un entero}
        write ( round ( pintas ) :4, ' ':3);

    FOR ale:= Mild TO Warmer DO
      BEGIN
        CASE ale OF
        Mild            : precio : =65;
        Bitter          : precio : =69;
        Special         : precio : =74;
        Warmer          : precio : =79
        END;     {CASE}
          {redondear y convertir a libras:}
        write ( round ( precio*pintas )
                          / 100 : 8 : 2 )
      END;
        {ahora empezar una nueva linea}
    WriteLn;
    pintas := pintas + 0.5

  UNTIL pintas >10
END.
```

¿A ti cuál te gusta?
El programa Real Ale imprime una lista de precios para cuatro clases distintas de cerveza para una gama de cantidades en incrementos de media pinta. Cada clase de cerveza se representa mediante un identificador de constante conceptual enumerado en la definición TYPE de cerveza. La selección del precio correspondiente se realiza naturalmente con una sentencia CASE. Las pintas enteras se imprimen como enteros rellenados con dos espacios extra, mientras que el formateo de las cantidades reales suprime las posiciones decimales no deseadas

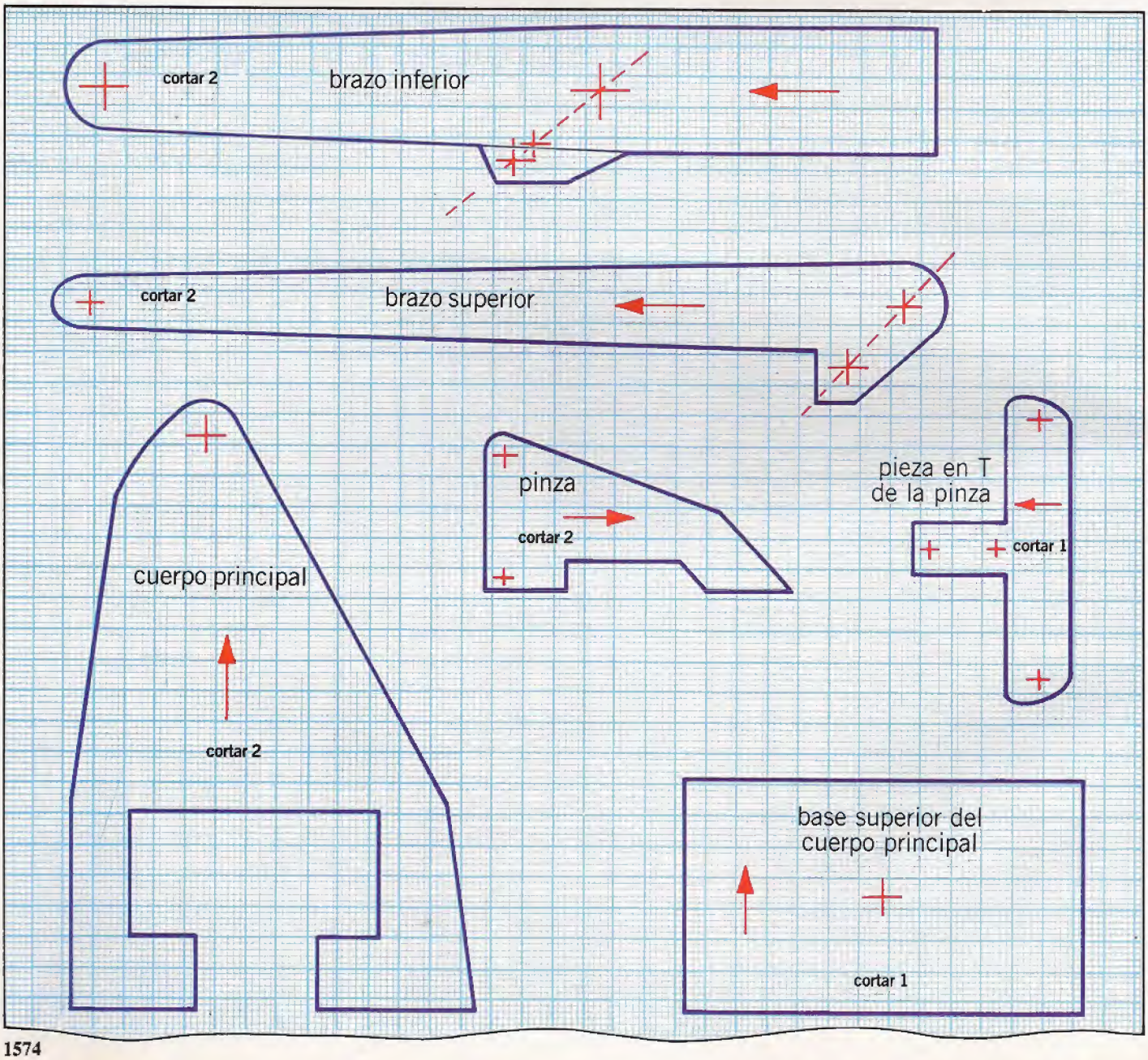
Un brazo fuerte

Iniciamos nuestro proyecto proporcionando los patrones de las partes del brazo y la lista de componentes

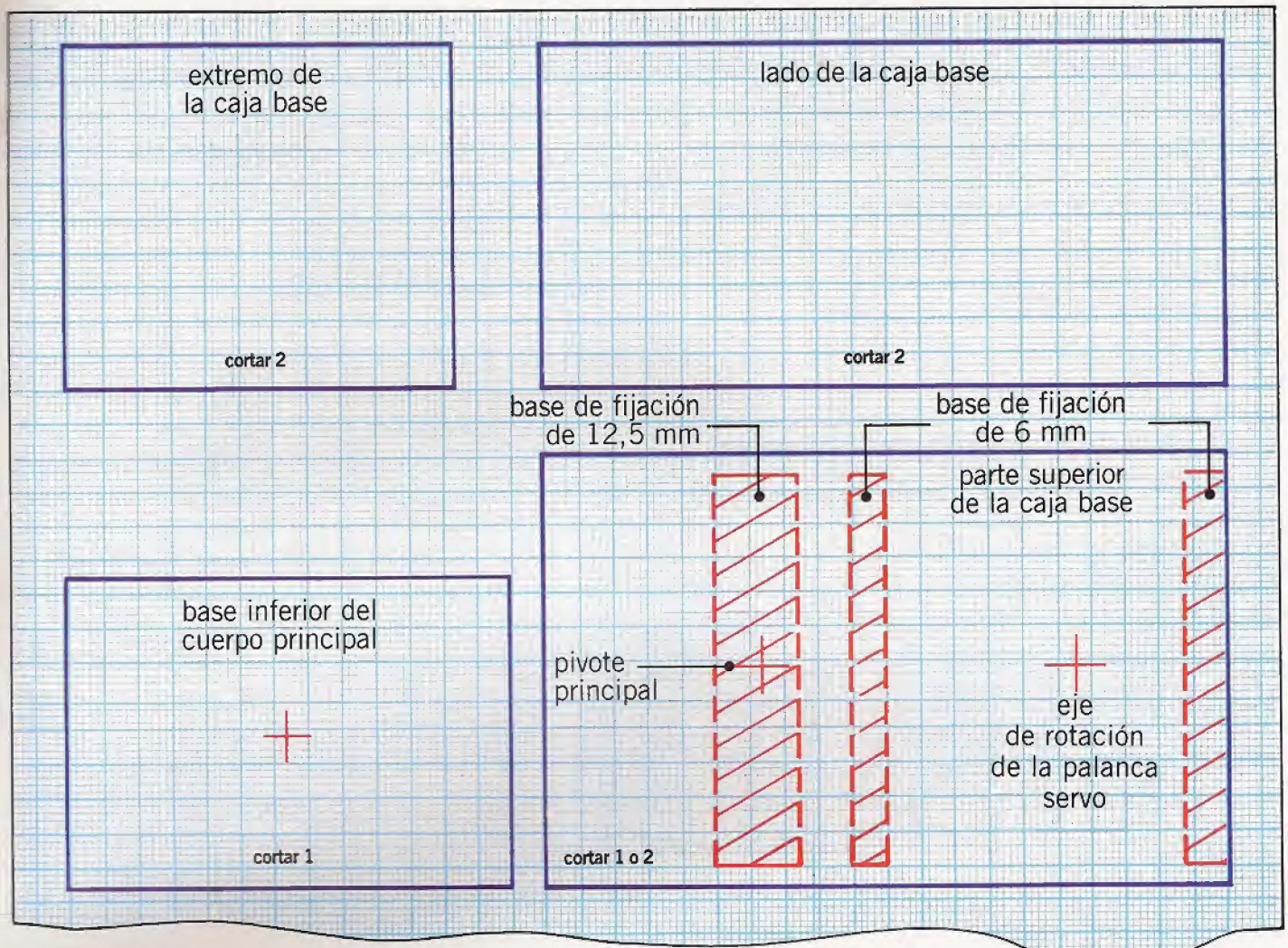
Los componentes necesarios para este proyecto probablemente habrán de obtenerse de diversas fuentes. Las tiendas de electrónica y de modelos a escala proporcionarán los componentes electrónicos y móviles, pudiéndose adquirir el resto en ferreterías.

La base y los miembros del brazo se deben construir con un material ligero y rígido que se pueda cortar y moldear fácilmente. Probablemente el material más accesible sea la madera contrachapada, aunque también se podría utilizar una lámina delgada de pérspex o un material plástico rígido. Suponiendo que para la estructura se emplee madera contrachapada, necesitaremos un cuadrado de 50 × 50 cm, de 4 mm de espesor. Lo usaremos junto con unos patrones para cortar los componentes de la estructura. Para actuar como bases de fijación necesitaremos tres trozos de madera de 55 mm (dos de 6 mm de espesor y uno de 12,5 mm de espesor).

Para el brazo necesitaremos cuatro servos digitales de 5 V; con este fin son ideales los motores Futaba S128. En el mismo sitio donde se adquieran los motores, se deben comprar discos plásticos de 30 mm para colocar en el extremo de los husillos



Kevin Jones



Cortado de los patrones
Haga calcos de los patrones para los componentes de la estructura, fíjelos sobre el tablero de contrachapado y córtelos. Observe que algunos patrones se usan dos veces para producir pares de componentes. Empareje los pares de los brazos y únalos antes de taladrarlos. Lije con cuidado cada uno de los componentes para eliminar las asperezas de los bordes

del motor. El mejor sitio tal vez sean las tiendas de modelos a escala. Si desea un brazo más potente, entonces habrá de emplear motores más potentes que el Futaba S128.

Para el motor necesitamos una fuente de alimentación eléctrica CD de 5 V. Aunque los tres micros que utilizaremos para controlar el brazo poseen fuentes de alimentación de 5 V en la puerta para ampliación o para el usuario, estos voltajes están diseñados para activar circuitos electrónicos externos a niveles TTL. La corriente máxima que se debe tomar de estas fuentes de 5 V es de 100 mA. Dado que un servomotor en carga máxima tomará 200 mA, no es aconsejable alimentar los servos del brazo desde la puerta para el usuario. Por consiguiente, necesitamos una fuente de alimentación externa. Lo ideal sería un transformador CD de 5 V, pero una alternativa menos onerosa es utilizar un conjunto unido de tres pilas de 1,5 V. La salida total de 4,5 V será suficiente para activar el sistema.

Se necesita un trozo de veroboard de 9 agujeros por 20 franjas para montar conectores en los que enchufar los motores, y dos metros de cable plano de 20 vías para conectar el brazo al ordenador. Quienes posean un BBC Micro habrán de comprar un conector IDC de 20 vías, y los que posean un Commodore 64 comprarán un conector marginal de 0,15 pulgadas y 24 vías para conectar el brazo a la puerta para el usuario. El brazo se unirá al Spectrum a través de la interface que hemos construido en *Bricolaje* para controlar el robot. En un próximo capítulo explicaremos en forma detallada las modificaciones necesarias en esta interface.

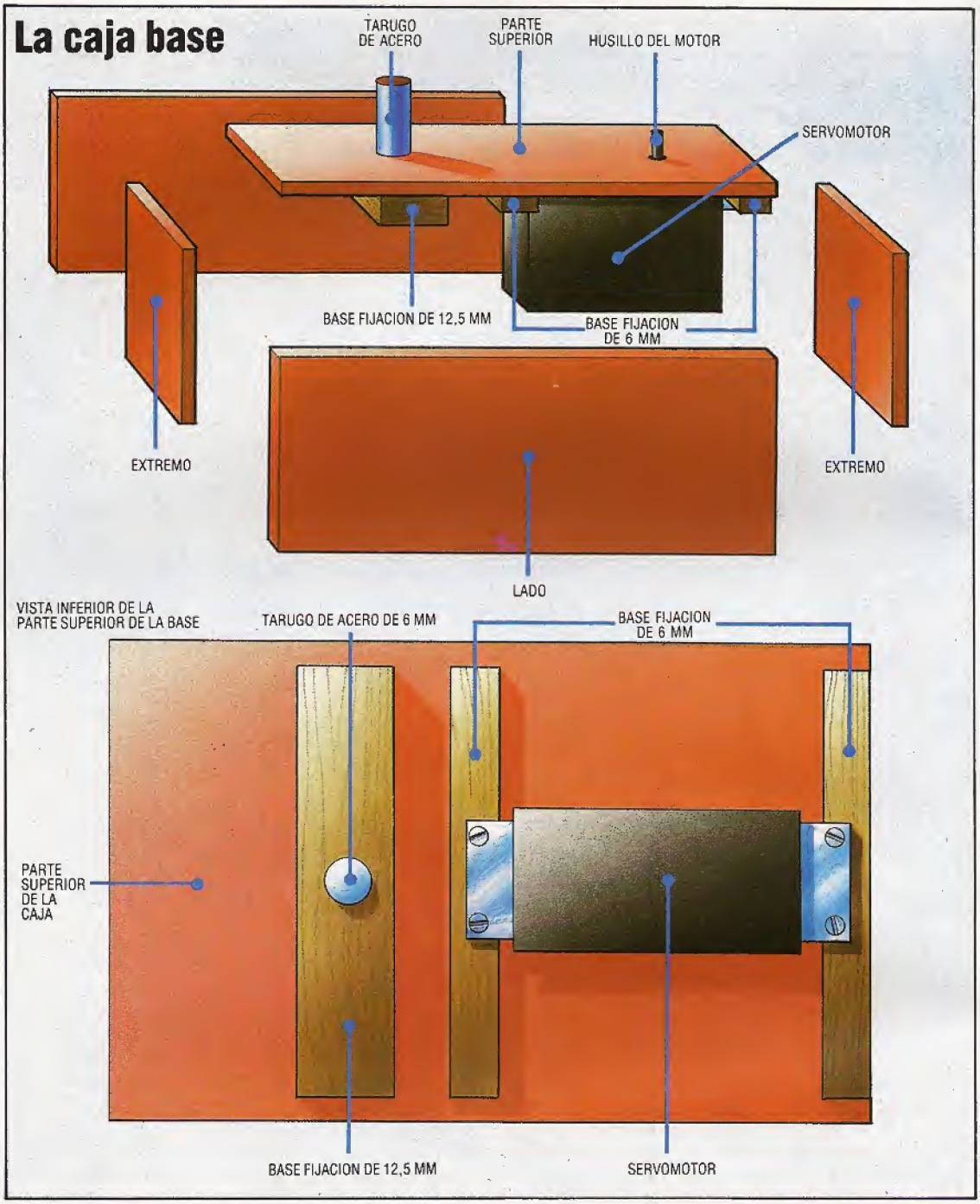
Para hacer los pivotes del brazo se requieren tres tubos de cobre de 75 mm de largo con un diámetro exterior de 4 mm, junto con tres trozos de tubo de cobre de 75 mm de largo y 5 mm de diámetro. Cada pivote se construirá colocando el tubo de 4 mm dentro del de 5 mm y recortando los extremos a la misma longitud. Por lo tanto, es importante que los tubos que compre se puedan colocar firmemente uno dentro del otro pero con suficiente amplitud como para que el tubo interior pueda girar libremente.

El montaje principal del brazo se une a la caja base mediante un cojinete de bolas. El cojinete debe estar rebordeado y tener un diámetro interno

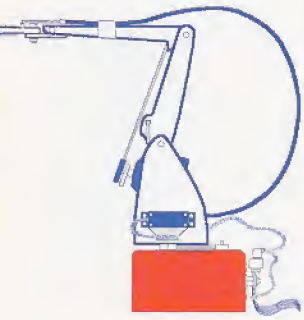
Lista de componentes

- Algunos tornillos para metales y tuercas 8 BA para montar la pinza
- 5 mm de separador de 6 mm de diámetro interno
- colas epoxy y para madera
- 16 tornillos de 10 mm sin tuerca y arandelas de goma para montar los motores
- 5 cm de espuma para las mandíbulas de la pinza
- Soldadura y cinta aislante

La caja base



Kevin Jones



de 6 mm. Estos cojinetes a menudo se utilizan para las ruedas delanteras de los coches a escala y se pueden conseguir en tiendas de modelismo. Para alargar el apoyo, también necesitaremos 25 mm de tarugo de acero de 6 mm. Este tarugo se fijará firmemente a la base y se encajará ajustadamente a través del cojinete.

La pinza se opera desde un motor montado más abajo en el ensamblaje del brazo. El motor debe conectarse a la pinza utilizando un enlace flexible de 500 mm de largo (denominado a veces *culebra*). En esencia, éste es un alambre de acero dentro de una manga de plástico. Los aviones a escala con frecuencia emplean este tipo de enlace para los alerones. Se necesitarán 50 mm de alambre duro (el cable de piano es ideal) para abrir y cerrar la pinza, y éste debe atarse al extremo de la culebra con un hilo de fusión a 5 A. Los miembros del brazo se mueven mediante un sistema de varillas. Para ello se utilizarán cuatro varillas de acero de 2 mm de grosor y 150 mm de longitud. Las varillas se pueden conectar a los discos del motor ya sea directamente o bien con pequeños cojinetes de bola.

Construcción de la caja base

Empiece por pegar con cola la base de fijación de 12,5 mm a la parte superior. Luego taladre los agujeros del pivote principal para alojar al tarugo de acero de 6 mm. Taladre atravesando la parte superior y la base de fijación, inserte el tarugo y péguelo bien en su sitio. Pegue las otras dos bases en su lugar, en la parte superior. Estas bases de fijación se utilizan para montar uno de los servomotores. Taladre el otro agujero de la parte superior de modo que a través del mismo pueda deslizarse el husillo, y atornille el motor sobre las bases, utilizando 4 tornillos sin tuerca y arandelas de goma. Pegue los extremos y el lado de la caja. Una vez secos, lije las esquinas y los bordes

Organizar los datos

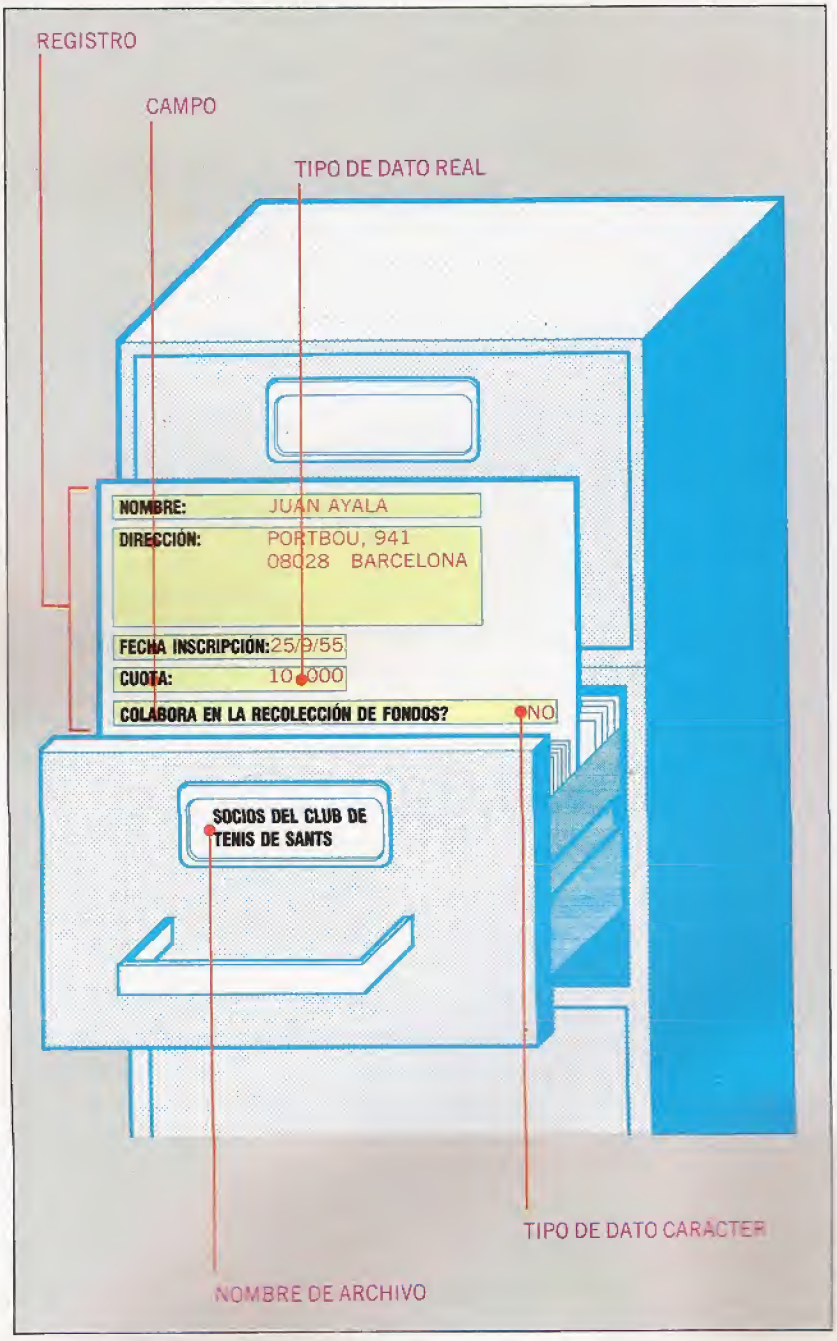
En este primer capítulo de esta nueva serie nos referiremos a las ideas en que debe basarse un sistema eficaz y bien estructurado

Suelen manipularse grandes cantidades de datos estructurados en entidades tan diferentes como pueden ser una secretaría de club, una agencia inmobiliaria o el departamento administrativo de un hospital. Un ordenador y un programa para administración de bases de datos pueden ser de ayuda para mantener organizada y actualizada la información. Pero antes de hablar de cómo se utilizan los programas de bases de datos, es importante pasar revista a los conceptos fundamentales que las mismas implican.

Para el programador de ordenadores, los datos se pueden considerar como estructurados o desestructurados. Los datos desestructurados son unidades aisladas, por lo general almacenadas en variables o constantes. Ejemplos de ello son NUMERO_DE_INTENTOS:=NUMERO_DE_INTENTOS+1 o IF ACIERTO=1 THEN PROCEDIMIENTO AUMENTAR. En estos dos ejemplos, NUMERO_DE_INTENTOS y ACIERTO son variables; en el primer caso, se está cambiando el valor de las variables, mientras que en el segundo se lo está comparando (con una posible bifurcación a un procedimiento). Es evidente que los objetivos NUMERO_DE_INTENTOS y ACIERTO sólo pueden tener un valor en un momento dado.

Los programadores de ordenadores, en especial quienes sólo están familiarizados con el BASIC, están acostumbrados a pensar en los datos de esta forma, como no estructurados; pero en la vida real todos estamos habituados a pensar en datos estructurados y a dar por sentadas dichas estructuras. Piense en sus amigos, por ejemplo. Ciertamente, en este sentido usted tiene su propia "estructura de datos". Conoce sus nombres, sabe su sexo, cuál es su edad (aproximadamente), tal vez sepa en qué trabajan y quizás incluso cuánto ganan, más otros detalles. A cada conjunto de datos relacionados, como éstos que hemos mencionado, lo llamamos *registro*. Cada registro se compone de uno o más campos (que pueden ser de tipos de datos similares o no), y el conjunto de registros se denomina *archivo*.

Para poner un ejemplo concreto, consideremos al secretario de un club de golf que tiene inscritos 500 socios. Para tener todo en orden, el secretario lleva un índice de fichas: una ficha para cada socio. Cada ficha contiene varias clases (tipos) de información (datos) acerca de un solo socio. La información podría incluir: APELLIDO, NOMBRE, SEXO, EDAD, AÑO DE INSCRIPCION, PAGO LA CUOTA?, SUELDO, NUMERO DE SOCIO, etc. En la terminología informática, cada uno de estos elementos de datos dentro del registro se denomina *campo*, y los diversos campos contienen diferentes tipos de datos. APELLIDO y NOMBRE serán, obviamente, series de caracteres, de modo que 137 4662 no sería un dato apropiado para el nombre de un socio. La entrada para SEXO sería un tipo booleano, capaz de tomar sólo uno de dos valores binarios. EDAD será un tipo entero (siempre que no deseemos saber que la edad de un socio es 37,224). AÑO DE INSCRIPCION será un valor entero con una gama limitada. Supondremos que AÑO DE INSCRIPCION = 1066 es inapropiado, aun cuando el archivo contenga registros de antiguos socios. Asimismo, también es impro-



Ian McKinnell

bable AÑO DE INSCRIPCION = 2001, a menos que el reglamento del club permita la inscripción anticipada de niños que se harán socios en el futuro. SUELDO sería un dato de tipo real; alguien podría tener un sueldo de 12 345,67 libras esterlinas.

Los programadores deben estar muy al tanto de los tipos de datos de que disponga el lenguaje de programación escogido. Algunos lenguajes, incluyendo al BASIC y al C, son pobres en cuanto a tipos, mientras que otros, como el PASCAL, insisten en que todos los elementos de datos se adecuen a tipos estrictamente definidos.

Si no tiene claro por qué esto es importante, veamos primero cómo tratan los datos el BASIC y el PASCAL, antes de pasar a definir una base de datos.

El BASIC sólo reconoce dos estructuras de datos: variables y archivos. Si en un programa escrito en BASIC se requiere una mayor estructuración de los datos, el programador habrá de imponerla, porque el lenguaje no la proporciona. Las variables del BASIC pueden ser de numerosos tipos diferentes, incluyendo valores enteros (MARCADOR%=7, p. ej.), reales de precisión simple (SUELDO!=1234.56), reales de doble precisión (SALARIO DE JOHN=123456.66666666666) y variables en serie (MARAVILLOSO$= Mi Computer). Como tipo de constante, en BASIC no existe nada parecido a PI=3.141592; por supuesto, uno siempre puede establecer el valor de la variable PI, pero igualmente puede incluir más adelante en el programa la sentencia PI=6.2.

Otros lenguajes de programación (y quizá el mejor ejemplo conocido sea el PASCAL) proporcionan varios tipos predefinidos y permiten que los programadores definan otros tipos nuevos, en términos de los tipos ya existentes. Los tipos predefinidos del PASCAL son constantes (no variables) y variables de tipo char (caracteres), integer (números enteros), real (números reales como 12.71) y boolean (datos binarios que son o verdaderos o falsos). Conjuntos, matrices, registros y archivos ya están todos incorporados, y el programador puede fácilmente definir nuevos tipos de datos. Veamos ahora cómo se podría crear el tipo de datos DIA.

```
TYPE
  DIA=(LUNES,MARTES,MIERCOLES,
       JUEVES,VIERNES,SABADO,DOMINGO);
```

Toda variable del tipo DIA que se utilice en el programa podrá tener sólo uno de los valores especificados, de modo que si se definiera DIA_LIBRE como del tipo DIA, DIA_LIBRE:=NOVIEMBRE sería ilegal, pero DIA_LIBRE:=DOMINGO se aceptaría.

Desde el punto de vista del programador de ordenadores, el tema de tipos estrictos tiene sus pros y sus contras; algunas veces es un obstáculo, otras veces ayuda a evitar errores de programación. Hemos cubierto algunos de los conceptos relativos a la definición de los tipos de datos, de modo que veamos ahora la importancia de las estructuras de datos en relación con las bases de datos.

¿Qué es una base de datos?

Una base de datos es un conjunto cualquiera de datos estructuralmente relacionados entre sí. Convencionalmente, los datos se organizan como un archivo (con su propio nombre de archivo), el archivo consta de registros (cada uno con su propio número de registro) y los registros están compuestos por uno o más campos de datos, conteniendo los campos uno o más tipos de datos.

Suponiendo que su base de datos está organizada como un índice de fichas de un fichero, la clave básica de cualquier entrada será, casi con toda seguridad, un nombre dispuesto por orden alfabético. Si existe la necesidad de buscar a un socio que se llame Martínez, usted irá pasando las fichas hasta hallar el archivo M, luego irá buscando con más detenimiento los apellidos que empiecen con Ma hasta que finalmente hallará las entradas que haya para Martínez. Sin embargo, si se le pidiera que buscara cuántos socios de sexo femenino tiene en el archivo, encontrar la respuesta sería laborioso. La tarea de hallar los datos llevaría aún más tiempo que si alguien deseara saber cuántos socios hay mayores de 36 años que ganen 11 000 libras al año.

Pero ésta es exactamente la clase de labor que puede hacer un programa de base de datos con sólo pulsar algunas de las teclas del ordenador. Hablando con exactitud, la base de datos es información, y el programa que ayude al usuario a entrar datos y manipularlos será lo que se denomina un *administrador de bases de datos*. Para mantener esta distinción entre el conjunto de datos y el programa que lo manipula, llamaremos al programa *administrador de bases de datos*, o DBM.

Las capacidades de los DBM varían considerablemente. Los más simples son poco más que agendas de direcciones electrónicas capaces de recuperar un registro cuando se les da un parámetro simple, como NOMBRE = ?. Los DBM más sofisticados poseen sus propios lenguajes de programación incorporados que permiten manipular la información de la base de datos de manera sumamente compleja, incluyendo la exportación de campos de datos a otros programas (como software de nómina de pagos, de facturación o de tratamiento de textos). En esta serie analizaremos los DBM, desde el más barato y sencillo hasta el más avanzado, y aprenderemos cómo usarlos de la forma más adecuada. Pero, no obstante, antes de hacerlo haremos una revisión sobre la tipología de los datos. Un buen DBM permitirá que el usuario defina el tipo de datos disponible para cada campo, y rechazará una entrada como NOMBRE =143326 o AÑO DE INSCRIPCION = 1066. Asimismo, dará un mensaje de error si usted intenta hallar registros en función de IF NOT (HOMBRE OR MUJER) AND SALARIO<0. Los buenos administradores de bases de datos, a diferencia de los lenguajes de programación, necesitan poseer una estricta comprobación de tipos para impedir que se entren datos erróneos cuando se están creando o modificando archivos.

En futuros capítulos analizaremos cómo organizan sus registros los diversos DBM. Consideraremos las bases de datos relacionales y de archivos múltiples, los campos y registros de longitud fija en comparación con los de longitud variable, creando bases de datos y extrayendo información de bases de datos. Para ilustrar los conceptos implicados, nos concentraremos en tres DBM: *Archive* de Psion, *dBase II* de Ashton Tate y *Card Box* de Caxton Software, si bien veremos someramente algunos otros paquetes.

Principales eventos en el BBC Micro

Los denominados "eventos" son señales de interrupción a las que la CPU puede permitirse no hacer caso

Hay numerosas situaciones en que las señales "eventos" son generadas en el BBC Micro: así, cuando se entran caracteres en el buffer de entrada, después de la generación de una señal sincrónica vertical, al terminarse una conversión en el convertidor A/D y cuando se pulsa la tecla Escape. Algunas de estas señales son ocurrencias de la máquina que también generan interrupciones, y son tratadas con el vector IRQV1.

Los eventos, pues, nos permiten aprovechar algunas de estas IRQ de máxima prioridad por medio de un método casi de "puerta de atrás". Para que se generen eventos y actúe en consecuencia el ordenador hay que "habilitarlos", o activarlos. Una vez activado, en cuanto sucede el evento particular, la CPU salta a la rutina cuya dirección de inicio se retiene en el vector en la dirección &220 y 221. Para que la rutina detenga su ejecución cuando se ha generado un evento, hay que *desactivar* éste. La activación y desactivación se hace por medio de dos llamadas OSBYTE. El siguiente cuadro muestra los eventos disponibles y las llamadas OSBYTE empleadas para su activación y desactivación. Aquí sólo damos los eventos más importantes y más comúnmente empleados.

Instrucciones para activar/desactivar eventos

Evento	Activar	Desactivar
Buffer de salida vacío	*FX14,0	*FX13,0
Buffer de entrada lleno	*FX14,1	*FX13,1
Ha entrado un car. en buffer entr.	*FX14,2	*FX13,2
Se ha completado conv. A/D	*FX14,3	*FX13,3
Inicio de sincronización vert.	*FX14,4	*FX13,4
Reloj de interv. se ha agotado	*FX14,5	*FX13,5
Se ha pulsado Escape	*FX14,6	*FX13,6

Así, por ejemplo, para activar el evento El reloj de intervalos se ha agotado ejecutaríamos simplemente *FX14,5 desde el BASIC, o su equivalente en lenguaje máquina. Después de activar este evento, la rutina cuya dirección se encuentra en el vector de eventos se ejecutará en cuanto el reloj de intervalos llegue a cero. Por esta razón, es importante que haya en esa dirección un fragmento de código máquina, de lo contrario puede ocurrir un crac.

Sin duda habrá notado que hay un único vector de eventos y, en cambio, toda una gama de eventos diversos. Y es que todos los eventos provocan un salto a la misma rutina, que debe por ello estar escrita de tal modo que les tenga en cuenta a todos ellos. ¿Cómo se determina cuál ha sido el evento causante de la entrada de la rutina? Si sólo se ha activado un evento, la respuesta es obvia; de otro modo hemos de buscar la respuesta en el contenido del registro A cuando se entra en la rutina. El cuadro siguiente muestra el contenido de varios registros a la entrada de la rutina apuntada por el vector de eventos.

Contenido de registros a la entrada de rutinas de eventos

Evento	Registro A	Registro X	Registro Y
Buffer de salida	0	Núm. del buffer	#
Vaciado del buffer de entrada	1	Núm. del buffer	#
Vaciado del carácter en buffer entr.	2	#	Carácter
Convertidor A/D	3	#	Núm. canal
Sincronización vertical	4	#	#
Reloj de intervalos	5	#	#
Condición Escape	6	#	#

En este cuadro, # indica que el contenido de registro no está especificado. Esto quiere decir que es posible, al entrar en la rutina de tratamiento del evento, comprobar el valor que está en el registro A de modo que se pueda determinar el evento causante de la entrada de la rutina.

Cuando usted escribe una rutina de tratamiento de eventos, debe, como primera tarea al entrar en la rutina, guardar el valor del registro. A diferencia de las facilidades de tratamiento de interrupciones en el BBC Micro, el tratamiento de eventos *no* guarda automáticamente el contenido del registro A y del registro de estado en la pila, por lo que tenemos que hacerlo nosotros. El tratamiento de eventos devuelve el control al programa interrumpido con una instrucción RTS. Dejamos aquí la teoría para pasar a examinar los eventos, uno a uno, con varios ejemplos.

- *Evento 0:* Se genera cuando se vacía el buffer de salida. El número del buffer empleado es el mismo que el usado por OSBYTE 21, que ya vimos al referirnos a la llamada OSBYTE.
- *Evento 1:* Se genera cuando se llena el actual buffer de entrada y se intenta introducir un nuevo carácter. El número especificado es el mismo que para OSBYTE 21. El código ASCII que está en el registro Y a la entrada del tratamiento del evento es el código que ya no cabe en el buffer.
- *Evento 2:* Generado siempre que entra un carácter en el buffer de entrada, por lo general después de ser pulsada una tecla. El siguiente programa muestra los elementos básicos de un tratamiento de eventos, generando un VDU 7 al pulsarse una tecla.

Esto es válido incluso cuando se pulsan teclas y el programa en BASIC se está ejecutando.

```
10 DIM mc% (100)
20 FOR I%=0 TO 2 STEP 2
30 P%=mc%
40 [ OPT I%
50 PHP
60 PHA
70 TXA:PHA
80 TYA:PHA
90
100 LDA # 7
110 JSR &FFE3
120
130 PLA:TAY
140 PLA:TAX
150 PLA
160 PLP
170 RTS
180
190 ]:NEXT I%
200 ?&220=mc%MOD256:REM byte inferior
    de la dirección colocado en el vector
210 ?&221=mc%DIV256:REM byte superior de la
    dirección
220 *FX 14,2
230 REM activa el evento sólo después de
240 REM establecida la rutina
250 REPEAT
260 PRINT I%
270 I%=I%+1
280 UNTIL FALSE
```

- *Evento 3:* Se genera cuando uno de los cuatro canales del convertidor A/D ha completado una conversión. Este evento tendrá lugar cada cinco o diez milisegundos, según la proporción de conversión escogida para el convertidor.
- *Evento 4:* Se genera al inicio de un impulso de sincronización vertical. Sucede 50 veces por segundo y puede emplearse como fuente de impulsos de reloj.
- *Evento 5:* Es probablemente el más útil de los eventos en el BBC Micro. Además de la variable TIME, el OS del BBC Micro actualiza un *reloj de intervalos* cada centésima de segundo. Ya tuvimos ocasión de analizar este reloj a propósito de ciertas llamadas OSWORD. El reloj es incrementado con regularidad y al llegar a cero se genera un evento. Es claro que esto nos permite interrumpir el proceso normal a intervalos regulares para que realice otra tarea concreta. El reloj de intervalos se escribe a través de la llamada OSWORD con A=4 y se lee por medio de OSWORD con A=3.

Si deseamos que el sistema genere un evento después de transcurrido un segundo, habremos de cargar el reloj con el valor máximo que puede representarse menos 100. Al incrementarse el reloj, cruzará el cero después de 100 tictacs. Dado que es una variable de cinco bytes, su valor máximo antes de tropezar con el cero es &FFFFFFFFFF. He aquí un programa que ilustra este evento, generando periódicamente un pitido:

```
10 DIM mc% (100), delay% 10
20 FOR I%=0 TO 2 STEP 2
30 P%=mc%
40 [ OPT I%
50 PHP:PHA
60 TXA:PHA
70 TYA:PHA
80 LDA # 7:JSR&FFE3
90 . clock set / restablece el reloj para la siguiente
   interrupción
100 LDX # delay% MOD 256 / establece el bloque de
    OSWORD
110 LDY#delay%DIV 256
120 LDA#4
130 JSR&FFF1
140 PLA:TAY
150 PLA:TAX
160 PLA:PLP
170 RTS
180 ]:NEXT I%
190 ?&220=mc%MOD256
200 ?&221=mc%DIV256
210 !delay%=&FFFFFF9C:delay%?4=&FF
220 *FX 14,5
230 CALL clock set: REM pone el reloj en marcha
240 END
```

- *Evento 6:* Se genera al pulsar la tecla Escape.

Hay además otros tres eventos a nuestra disposición, y suceden si se detecta un error en el sistema RS423, un error en el Econet o un Evento Especificado por el Usuario. Con esto damos por concluido el repaso de los eventos más útiles empleados por el OS del BBC Micro. El próximo capítulo ofrecerá una visión general de cómo actúa el sistema operativo al completo.

Ejercite sus conocimientos

El siguiente programa muestra algunos de los principios examinados hasta aquí en nuestro curso sobre el OS del BBC Micro. Examine cuidadosamente el listado, entre el programa y ejecútelo. Trate de adivinar: 1) qué operación está realizando el programa; 2) cómo funciona éste. En el próximo capítulo se dará una cumplida explicación de él, junto con otros programas de muestra. Una pista: el nombre de la variable en la línea 80 puede orientarle por el buen camino

```
 20 PROCassemble
 30 END
 50 DEFPROCassemble
 70 DIM MC% &FF
 80 oldvec=?&20E+256*?&20F
 90 FOR pass=0 TO 3 STEP 3
100 P%=MC%
120 [OPT pass
130 . temp EQUB 00
140 . start
150 PHP
160 STA temp
170 TXA:PHA
180 TYA:PHA
200 JSR check
230 PLA:TAY
240 PLA:TAX
250 LDA temp
260 PLP
270 JMP oldvec
290 . check
300 LDA temp
310 CMP#65
320 BMI out
330 CMP #91
340 BPL out
350 ADC #32
360 STA temp
370 .out RTS
390 ]:NEXT
400 ?&20E=start MOD 256
410 ?&20F=start DIV 256
420 ENDPROC
```

Datos didácticos

La utilización de bases de datos les permite a los niños desarrollar una notable destreza en el tratamiento de información

La obtención de información de una "biblioteca electrónica" reporta muchos beneficios educativos: el acceso es inmediato, y para recuperar datos no es necesario salir del aula, ni siquiera levantarse del asiento. Se evita la posibilidad de que alguien "se lleve el libro a casa". Además, los datos informatizados se pueden actualizar fácilmente, mientras que la información retenida en los libros de texto es estática y sólo se puede modificar mediante la reimpresión. Los costos actuales de la producción de libros representan un grave problema para la provisión de material fuente, y con frecuencia las escuelas se ven obligadas a proporcionar publicaciones anticuadas. Y, lo que es aún más significativo, el elevado coste de la palabra impresa puede impedir, asimismo, la difusión de nuevos métodos educativos. En una clase dotada de ordenadores, para modificar los programas de estudio sólo es necesario cambiar los discos maestros: el reaprovisionamiento completo de la biblioteca se vuelve innecesario.

Los beneficios educativos se hacen especialmente evidentes mediante la combinación del microordenador y las telecomunicaciones para servicios de base de datos en línea. Tener, por ejemplo, la Enciclopedia Larousse en disco sería una aspiración sumamente útil. Si la información que usted deseara se pudiera hallar buscando en los índices comunes, por orden alfabético, entonces sería perfectamente adecuado tomar un ejemplar del libro de la estantería. Pero si deseara hallar todos los marsupiales *de madriguera*, por ejemplo, o confeccionar una lista de todos los países donde el porcentaje del PNB invertido en educación esté por encima de un cierto nivel, entonces compilar estos datos a mano a partir del texto sería una tarea agotadora.

La Enciclopedia Larousse, sin embargo, ocupa

Un mundo para ellos
El temor de que el uso de ordenadores en la escuela confinara a los niños al estrecho mundo de la VDU parece totalmente infundado: la práctica ha demostrado que el software para bases de datos, en particular, puede inducir al niño a realizar todo tipo de actividades ajenas al teclado. La necesidad de reunir información para entrar en la base de datos se puede abordar de muchas formas diferentes, incluyendo la de "caminar por la naturaleza", en la cual los estudiantes son estimulados para observar, registrar y clasificar elementos del medio ambiente

Steve Cross

muchos volúmenes y el espacio de almacenamiento y las amplias facilidades de búsqueda necesarias requerirían una potencia de proceso extremadamente grande; y es aquí donde cobran importancia los servicios en línea. El usuario entra en ellos con un modem estándar de 300 o 1 200 baudios y la línea telefónica, para acceder a una cantidad abrumadora de información. La base de datos Inspec, por ejemplo, compilada por el Institute of Electrical Engineers, cubre toda la gama de ingeniería eléctrica y electrónica, con más de dos millones de entradas. La Royal Society of Chemists produce Chemical Engineering Abstracts, donde se listan trabajos de investigación, artículos y otros materiales de referencia. Si bien para mantenerse al día de la investigación vale la pena disponer de las versiones impresas, es más fácil buscar las referencias específicas en línea.

Dialog, en Estados Unidos, es el mayor servicio de base de datos en línea de todo el mundo y uno de los más utilizados en el campo de la educación. Contiene 200 bases que cubren más de 100 publicaciones y 100 millones de elementos de información: extractos de todo, desde historia a zoología. Cuando usted busca en el Dialog, se le ofrecen referencias no sólo del autor y el título de cualquier informe o artículo relevantes, sino que también se le ofrece un extracto con una descripción detallada y, con frecuencia, conteniendo toda la información que necesite. Por supuesto, se puede obtener una salida impresa o bien pedir que la información impresa se le envíe, por lo general a un costo moderado. La cantidad de bases de datos en línea disponibles se está respaldando con una creciente cantidad de software para búsqueda en bases de datos, para ayudar a los usuarios a extraer y clasificar la información.

En su *Schools link*, el Prestel ofrece un enfoque diferente. A través de este servicio se les ofrece a las escuelas información sobre todos los aspectos de la informática y se crea un foro de ideas y contactos. Está dirigido a los maestros en lugar de a los alumnos, y contiene comentarios de software, robots educativos, libros y sugerencias para emprender proyectos; asimismo, facilita programas educativos para cargar en la máquina del maestro o de la escuela.

No obstante, además de la "biblioteca electrónica", las bases de datos están adquiriendo creciente popularidad en la clase, donde juegan un papel muy activo. Ya existen varios programas que le ofrecen al niño en edad escolar la experiencia de manipular información y estructurarla de acuerdo a sus propias necesidades individuales, proporcionándole al niño la oportunidad de reunir, entrar y manipular sus propios datos.

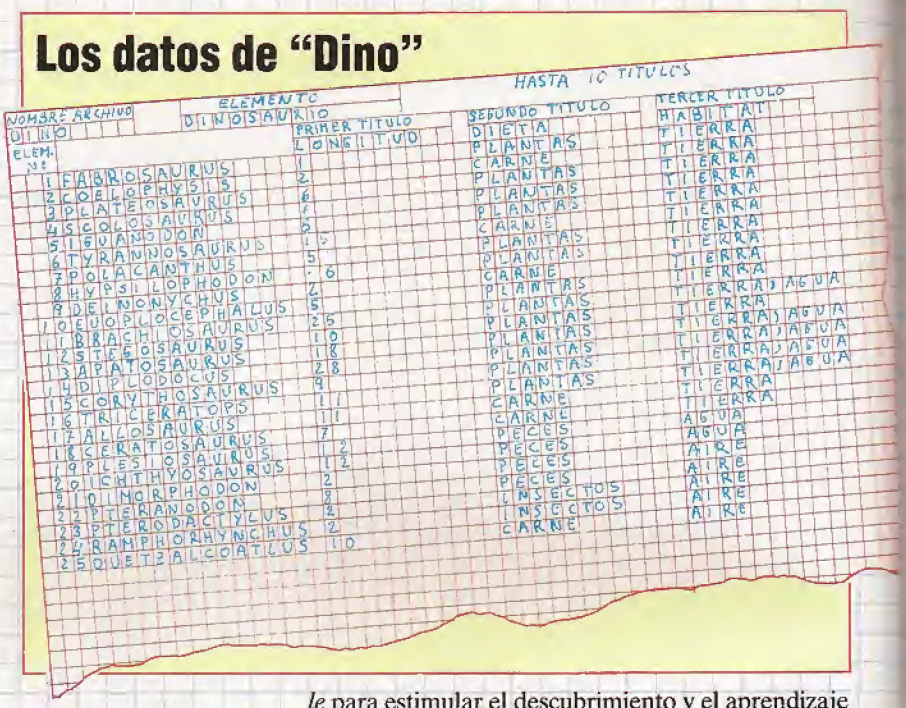
Una base de datos llamada "Factfile"

Las posibilidades del uso de las bases de datos fueron el tema de un congreso educativo que tuvo lugar en 1981 en la Universidad de Cambridge, en Gran Bretaña, y el debate inspiró la creación de un programa llamado *Factfile* (Archivo de hechos), que permite a los niños pequeños construir archivos de datos sobre cualquier tema que elijan.

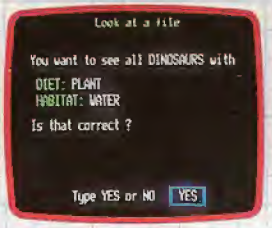
El papel activo de una base de datos como *Factfile* para estimular el descubrimiento y el aprendizaje se hizo evidente enseguida. Supongamos que los niños desean crear una base de datos que contenga información sobre sus maestros. Primero tendrán que pensar en las categorías por las que van a organizar la información. Esto dependerá de lo que ya sepan y de lo que puedan descubrir por sí mismos. Podrían decidir, por ejemplo, incluir edad, altura, asignatura, sexo y "tipo de maestro". Habiendo determinado la estructura de la base de datos, deben luego reunir información para las categorías específicas. En este punto, el ejercicio se convierte en una operación educativa variada: como mínimo, los niños tendrán que reunir información física. Si se hubieran de realizar mediciones, deberán determinar las unidades a utilizar y los datos a incluir. El proceso adquiere mayor complejidad puesto que el niño comienza a comprender el significado de las categorías que ellos mismos han generado. El campo "tipo de maestro" exige un buen nivel de análisis: significa "¿Qué asignatura enseña?" o "¿Es el maestro agradable o antipático?".

Cuando hayan terminado de compilar la base de datos, otros niños pueden utilizarla, correlacionando información de más de una categoría. Quizá deseen listar todas las maestras que se han clasificado en la categoría "agradable" o todos los maestros de matemáticas. El proceso de aprendizaje da otro paso más hacia adelante cuando el niño empieza a analizar los resultados. *Factfile* permite que los niños guarden su base de datos en disco o cassette y añadan información posteriormente, permitiéndoles retornar a los archivos siempre que así sea necesario.

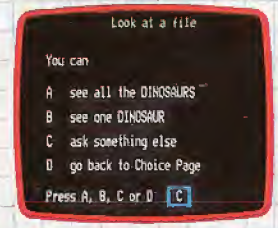
Una crítica que se le suele hacer al uso del ordenador por parte de los niños es que el medio los estimula a permanecer sentados delante de la VDU desarrollando actividades que no guardan relación con el mundo en el que viven. Pero *Factfile*, aparte de facilitarle al alumno una experiencia directa con bases de datos, favorece numerosas actividades ajenas al teclado. Si se crea un archivo con detalles de la fauna local, los niños tendrán que registrar cuidadosamente los detalles de cada planta y después de-



Hace un millón de años...
Factfile es un sistema completo de base de datos para los estudiantes más jóvenes, que representa una excelente introducción a la administración de bases de datos. Con el programa se entrega un archivo muestra, llamado "Dino", que contiene información sobre los dinosaurios. Se estimula a los niños para que se familiaricen con el "Dino" y cataloguen el contenido del archivo (que vemos arriba) antes de pasar a crear sus propias bases de datos

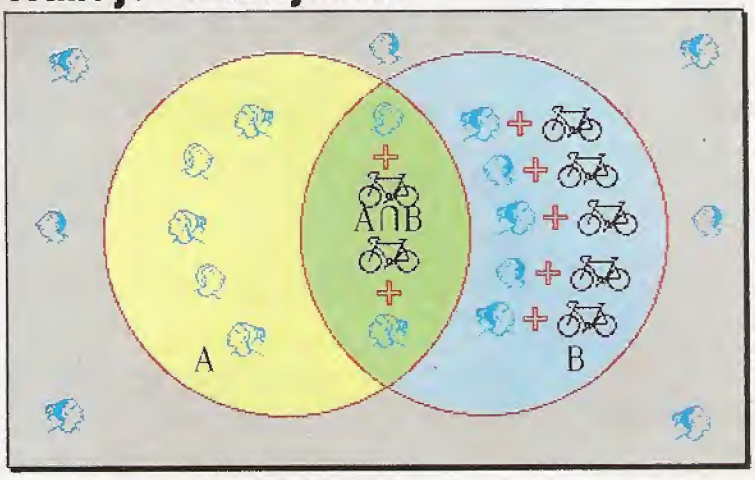


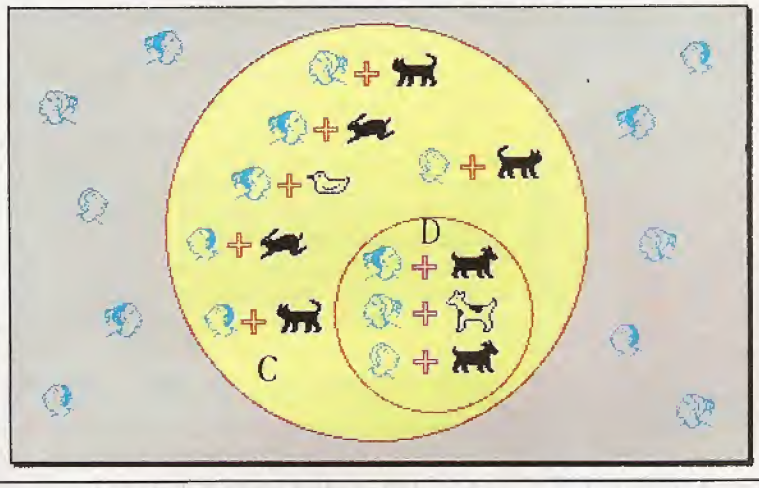
Solicitud de búsqueda



Menú del "Dino"

Trabajo de conjuntos





Ian McKinnell

sarrollar su propio sistema de clasificación al objeto de construir la base de datos. En una escuela de Londres se construyó una base de datos como parte de un programa de estudios sobre el medio ambiente, utilizando como entrada los resultados de una encuesta efectuada entre personas de aquella zona de la capital británica.

Your facts (Tus hechos), diseñado para niños más pequeños, es otro interesante paquete. Al pequeño se le pregunta su nombre, si es niño o niña, si tiene algún animalito, un reloj, una bicicleta y un hermano o hermana. Después de digitarse la información concerniente a un alumno, el programa pregunta si a alguien más le gustaría digitar su propia información. Hay lugar para datos de unos 40 niños, por lo cual se puede incluir a toda la clase. Una vez digitada toda la información, y tras haberse digitado "No" en respuesta a "¿ALGUIEN MAS QUIERE HACER UNA ENTRADA?", los niños retornan al menú, donde pueden ver todos los registros, descubrir qué niños caen en una categoría determinada (p. ej., quiénes tienen un reloj o quiénes tienen una bicicleta), o bien pueden participar en un juego. Éste consiste en que el ordenador adivina el nombre de un niño tras formular varias preguntas como "¿ERES UN NIÑO?", "¿TIENES UN RELOJ?", etc. Luego pregunta, por ejemplo, "¿TE LLAMAS TIMOTHY?".

Your facts constituye para el niño pequeño una introducción ideal al concepto de lo que es una base de datos. Fomenta la lectura y la escritura, establece con el niño una relación personal y es un ejercicio entretenido.

Información clasificada

La construcción y la interrogación de bases de datos guarda cierta relación con otras tareas que realizan los niños en la escuela. Los programas de estudio de matemática moderna introducen la noción de conjunto a una edad muy temprana. La teoría de conjuntos guarda una estrecha relación con la clasificación de datos y las relaciones entre tales clasificaciones. Preparar una base de datos es, en muchos sentidos, una labor directamente análoga, categorizando las propiedades de una entrada en campos y registros.

Los conceptos de intersección y subconjunto, en especial, poseen un verdadero significado en la interrogación de bases de datos, correspondiendo a la creación de listas cortas. La intersección es la "superposición" de dos conjuntos: por ejemplo, la intersección de un conjunto de niños rubios con el conjunto de niños que tienen bicicleta es el conjunto de niños rubios que poseen bicicleta. Un caso especial de intersección es aquel en el que un conjunto se superpone completamente con otro, formando un subconjunto, como en el segundo ejemplo

El programa "One world"

Una empresa de software australiana, Active Learning Systems, ha producido un amplio paquete de base de datos de información relativa a casi todos los países del mundo. El nombre del programa es *One world* (Un solo mundo) e incluye observaciones y hojas de trabajo docentes para el individuo y para la clase. Contiene más de 30 elementos de información para cada país, incluyendo datos sobre el sistema político, importaciones, exportaciones, idiomas, países limítrofes, religión, tasa de alfabetismo, antecedentes históricos y tratados. Se da el porcentaje de población rural y urbana, así como porcentajes de la fuerza de trabajo en la producción, manufactura y principales industrias de servicios. También se incluyen las proporciones de zonas desérticas, bosques y tierra cultivada. El atractivo del programa, al igual que el de la mayoría de las bases de datos, reside en sus facilidades para buscar y analizar los datos.

El menú principal ofrece al usuario una serie de opciones:

1. VISUALIZAR CUALQUIER PAIS
2. BUSCAR UTILIZANDO INFORMACION UNICA
3. ANALIZAR UTILIZANDO DIVERSOS CRITERIOS
4. REFERENCIA DE AYUDA
5. SALIR

Si un niño conociera la existencia del señor Shamir, un líder del Oriente Medio, y quisiera averiguar su país, seleccionaría la segunda opción. El menú BUSCAR DATO permitiría que el niño digitara su nombre en 3.JEFE DE GOBIERNO y 4.JEFE DE ESTADO. El programa le informaría entonces que YITZHAK SHAMIR ocupaba el cargo de jefe de estado de ISRAEL.

La opción ANALIZAR es la más brillante del programa. Permite que el niño analice información, utilizando hasta tres opciones, entre 20 criterios. Por ejemplo, podría averiguar en qué países europeos la mayoría de la población vive en el campo. Ello sólo emplearía dos opciones: región y porcentaje de población. Si el niño se entera de que los mineros de bauxita han ido a la huelga en Surinam, rápidamente puede descubrir la situación del país, que la bauxita es una de las principales exportaciones y que la huelga puede tener un efecto devastador en la economía.

One world enseña las técnicas de la investigación e interpretación de datos, y estimula a los niños a sacar conclusiones a partir de los análisis que hagan de la información suministrada. Constituye un gran apoyo para las lecciones de historia, geografía y ciencias sociales.

Una base de datos puede indicarle al alumno la mayoría de lo que hay para aprender y lo estimula a buscar las respuestas. El niño puede observar cómo manipula la información el ordenador y comprender la importancia de formular las preguntas correctas y organizar adecuadamente las entradas. Quienes, tras haber adquirido experiencia en el uso de una base de datos, accedan a la educación superior y finalmente al mercado de trabajo, con toda seguridad poseerán una destreza verdaderamente importante.

Nombre por nombre

En esta ocasión construiremos un sistema sencillo basado en la idea de una agenda de direcciones

Una de las principales ventajas de la agenda de direcciones tradicional es que, a excepción de la organización por orden alfabético, hay muy poca estructuración. Es flexible en el sentido de que uno puede tener entradas tales como:

PEDRO FERNÁNDEZ, Pedreña 809, Madrid (91-3397881)
— Paloma, su novia — despacho: 91-3388771 ext 160
— llamar después de las 4 — en Ene. se muda a un piso nuevo.

Seguida de:

FIDEL — 396-1949

Seguida de:

FINA — ver Bermúdez

Utilizando una agenda de direcciones, uno está en libertad de incluir a las personas por su nombre de pila, su apellido o incluso mediante "direccionamiento indirecto", con "apuntadores" a otras entradas. Sin embargo, cuando una base de datos como ésta se transfiere a un ordenador, por lo general se requiere un enfoque más sistemático.

Dado que, en comparación con la original, una agenda de direcciones informatizada posee algunas desventajas, vamos a diseñar el formato para una que sea lo suficientemente flexible como para hacer frente a la mayoría de las circunstancias. Consideraremos cada campo uno por uno, empezando por el del nombre.

El nombre tiene dos partes: la mitad genérica, denominada apellido, y la mitad específica, denominada nombre de pila. En la mayoría de los países, el o los nombres de pila van seguidos por el apellido, mientras que en China, Japón, Hungría y otros países, el apellido va antes del nombre de pila. En Japón, hay sólo un nombre de pila, mientras que en la comunidad china de Inglaterra se acostumbra a tener un nombre de pila inglés además del nombre de pila chino. De este modo, Li (apellido) Yu Chow (nombres de pila) para la mayoría de sus amigos ingleses será Paul.

Con esto ya tenemos muchísimas probabilidades de que se produzcan confusiones, y eso que sólo estamos teniendo en cuenta el campo del nombre. Es evidente que hemos de imponer alguna disciplina y decidir qué irá primero, si los nombres de pila o los apellidos, y qué longitud podrá concederse a un nombre. Los nombres pueden ser tan cortos como Ng o tan largos como Cholmondley-Smythe, y posiblemente incluso más cortos o más largos, de modo que siempre debemos dar cabida a los casos más extremos. Si el administrador de la base de datos (DBM) utiliza campos de longitud fija, tendremos que elegir una longitud de campo adecuada para el nombre más largo con el que podamos encontrarnos. (Más adelante, en esta serie, hablaremos de las desventajas que suponen los campos de longitud fija.)

Si hemos de imponer un formato, como en realidad debemos hacer al diseñar una base de datos, probablemente lo más simple sea utilizar el campo del apellido como el campo clave fundamental, de modo que comenzamos con el apellido, que irá seguido por uno o más nombres de pila. La dirección, asimismo, plantea problemas. Indudablemente, usted habrá visto en revistas norteamericanas formularios para recortar que piden su Ciudad, Estado y Código Postal. Las direcciones en España no se ajustan a ese formato. Otra complicación surge al considerar que incluso el formato "Nombre, Número de la Casa, Calle, Ciudad, Provincia, País" tampoco es apropiado. En Japón, por ejemplo, la ciudad ocupa el primer lugar en una dirección, seguida por el distrito de la ciudad, seguida por el número del solar del edificio dentro del distrito, viniendo en último lugar el nombre del destinatario.

Una especificación básica

En este sencillo ejemplo de base de datos, al igual que en las más complejas, probablemente no sea posible hacer frente a todas las combinaciones que se puedan concebir, de modo que habremos de llegar a un compromiso. Suponiendo que no es probable que tenga que comunicarse alguna vez con alguien que tenga 16 nombres de pila, veamos cómo funcionaría el siguiente esqueleto básico:

- Campo del Apellido: hasta 40 caracteres
- Campo del Nombre de Pila: hasta 60 caracteres
- 1.ª Línea Campo de Dir.: hasta 80 caracteres
- 2.ª Línea Campo de Dir.: hasta 80 caracteres
- 3.ª Línea Campo de Dir.: hasta 80 caracteres
- 4.ª Línea Campo de Dir.: hasta 80 caracteres
- 5.ª Línea Campo de Dir.: hasta 80 caracteres
- Campo del Teléfono: hasta 20 caracteres
- Campo de Notas: hasta 80 caracteres

Estas especificaciones básicas cubrirán la mayoría de las eventualidades, aunque aún podrían plantearse problemas. Una dificultad potencial es la limitada longitud del Campo de Notas. Otro problema es que País no está especificado como un campo separado. Según la longitud de la dirección, el país puede hallarse en el tercero, cuarto o quinto campo de la dirección, o en ninguno en absoluto. Ello normalmente no representaría ningún problema hasta que deseáramos buscar en nuestra base de datos utilizando País como clave. Si su aplicación de base de datos ha de tratar con muchos corresponsales extranjeros, será mejor diseñar la base con un campo de País separado. Decisiones de esta clase siempre serán necesarias en la etapa de diseño.

El DBM que ejecute en su ordenador determina-

rá la facilidad con la que pueda implementar una base de datos en su sistema. Uno de los DBM más simples es el *Card box* de Caxton. El programa posee facilidades limitadas para extraer y manipular datos, pero si sus exigencias son normales, le dará los resultados que necesite con un mínimo trabajo. *Card box* no ofrece un lenguaje de programación sofisticado para manipular campos ni registros. Sí permite, sin embargo, que el usuario extraiga registros específicos entrando desde el teclado instrucciones simples. Utilizando el *Card box* para crear la base de datos esbozada más arriba, usted sólo habrá de cargar el programa y seguir una sencilla secuencia de entradas.

Uso del "Card box"

Lo primero que debe hacer es decidir el formato del registro. Este formato determina qué información se almacenará, cómo se indexará (es decir, qué campos se designarán como campos "clave") y cómo aparecerán finalmente los registros en la pantalla del ordenador o en las salidas impresas. Si al archivo de la base de datos lo llamamos AGENDADIR, *Card box* creará un archivo de formato denominado AGENDADIR.FMT. Éste se puede editar si fuera necesario modificar la forma en que se visualiza la información. También se pueden crear archivos de formato alternativos para visualizar la información de la base de datos de formas diferentes.

Al igual que la mayoría de los DBM, *Card box* permite entrar texto *permanente* en cada uno de los campos. En el caso de una agenda de direcciones, es bastante evidente cuál es el significado de cada campo: Juan Pérez es a todas luces un nombre y no parte de una dirección; de modo similar, 339-44-20 es, obviamente, un número de teléfono. En otras bases de datos, quizá deba recordar cuál es el significado de cada campo. Es allí donde entra en juego el texto permanente. Compare estos dos registros:

```
06116
3995
86
34.75
Paquete 100 folios
```

y

```
NUMERO DE COMPONENTE DEL FABRICANTE 06116
NUMERO DE COMPONENTE NUESTRO        3995
CANTIDAD EN STOCK                   86
PRECIO                              34.75
DESCRIPCION                         Paquete 100 folios
```

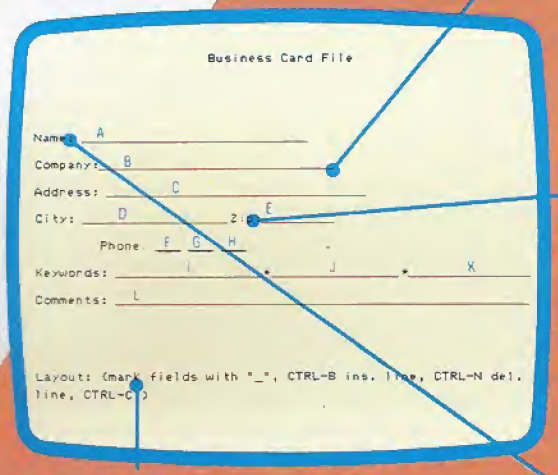
La información es idéntica en ambos casos, pero en el segundo ejemplo, con la aparición de un texto permanente en cada registro, es mucho menos probable que se produzcan errores.

Card box permite dar a cada campo uno de cuatro atributos posibles: NONE (ninguno), MAN(ual), AUTO o ALL (todos). En una base de datos de stock, es improbable que alguna vez sea necesario usar PRECIO como campo clave: no es probable que usted pregunte: "¿Tiene algún componente que cueste menos de 600 ptas. y más de 400 ptas?" No obstante, es bastante posible que necesitemos saber cuántos artículos #06116 hay todavía en stock, de modo que será necesario que NUMERO DE COMPONENTE DEL FABRICANTE sea un campo clave.

Retomando nuestro ejemplo de la agenda de direcciones, es seguro que desearemos buscar en la base de datos en función de Nombre y, probablemente, también de Nombre de Pila. Ambos habrán de ser campos clave. Si estuviéramos llevando una empresa, podríamos necesitar, asimismo, buscar registros por Ciudad y posiblemente incluso por Código de Distrito Telefónico. Es importante recordar que no se puede crear una base de datos eficaz a menos que uno haya diseñado perfectamente cómo habrá de emplearse la base. A diferencia de los ficheros de tarjetas tradicionales, las bases de datos exigen que uno prevea de antemano cómo se la utilizará en el futuro.

Elementos de diseño

Megafinder permite al usuario diseñar sus propios formularios. Éste se obtuvo a partir del mismo formulario que se proporciona ya hecho con el programa, que es el que utilizamos en nuestra otra visualización

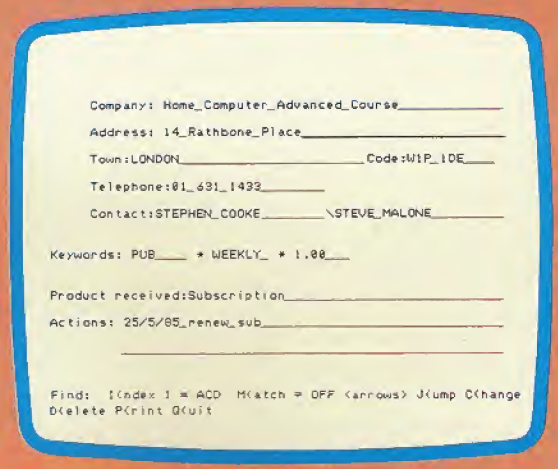


LONGITUD DE CAMPO
Se indica mediante caracteres de subrayado (_). La longitud total de todos los campos de un formulario determina la cantidad de formularios que se pueden almacenar en un archivo. Por consiguiente, resulta práctico mantener los campos tan cortos como sea posible, teniendo presente la naturaleza de los datos a entrar

ETIQUETAS DE CAMPO
Se las entra al diseñar el formulario; en este programa se incluyen sólo para comodidad del usuario. El programa las emplea cuando visualiza información en la pantalla, pero no tiene en cuenta las etiquetas de los campos cuando clasifica datos, tarea que realiza utilizando IDENTIFICADORES DE CAMPO

IDENTIFICADORES DE CAMPO
Los identificadores de campo, que no se deben confundir con las ETIQUETAS DE CAMPO, determinan el orden por el cual se accederá a los campos cuando se entre información. Asimismo, los identificadores permiten clasificar los campos por separado o en grupos. Utilizando la opción INDEX (índice) de *Megafinder*, la asignación del campo D al Indice 1 le permitiría al usuario clasificar rápidamente los diferentes registros de acuerdo a la disposición alfabética de ciudades

LÍNEA DE INSTRUCCIONES
Muestra algunas de las instrucciones que se pueden utilizar al editar un diseño. Se pueden insertar y suprimir líneas y caracteres, y el diseño terminado se puede guardar en disco pulsando CRTL-C



Permítame que le enseñe mi tarjeta
Este vuelco de pantalla del *Megafinder*, un DBM que se ejecuta en el Apple, muestra un registro seleccionado de un archivo llamado "Tarjetas de empresas" y, debajo del mismo, una lista de instrucciones opcionales. Los datos retenidos por el programa se pueden clasificar en hasta cuatro índices diferentes. El DBM también puede buscar una pareja (MATCH) entre los datos entrados y los almacenados, saltar (JUMP) a secciones especificadas de la base de datos (p. ej., a la que retiene nombres de empresas que empiezan con M), cambiar (CHANGE) y suprimir (DELETE) información, e imprimir (PRINT) un registro o un listado

Levando anclas

Finalizados los preparativos, ya podemos zarpar y desarrollar un módulo que se encargue del progreso semanal de la travesía

Las incidencias semanales durante el viaje se pueden controlar desde un simple bucle FOR...NEXT, utilizando un contador de bucle, WK, para contar las semanas del viaje. El límite superior del bucle se establece mediante el valor de la variable JL, cuyo valor al comienzo del programa se establece en ocho. Desde el bucle podemos llamar a subrutinas para tratar cada una de las tareas que se produzcan sobre una base semanal.

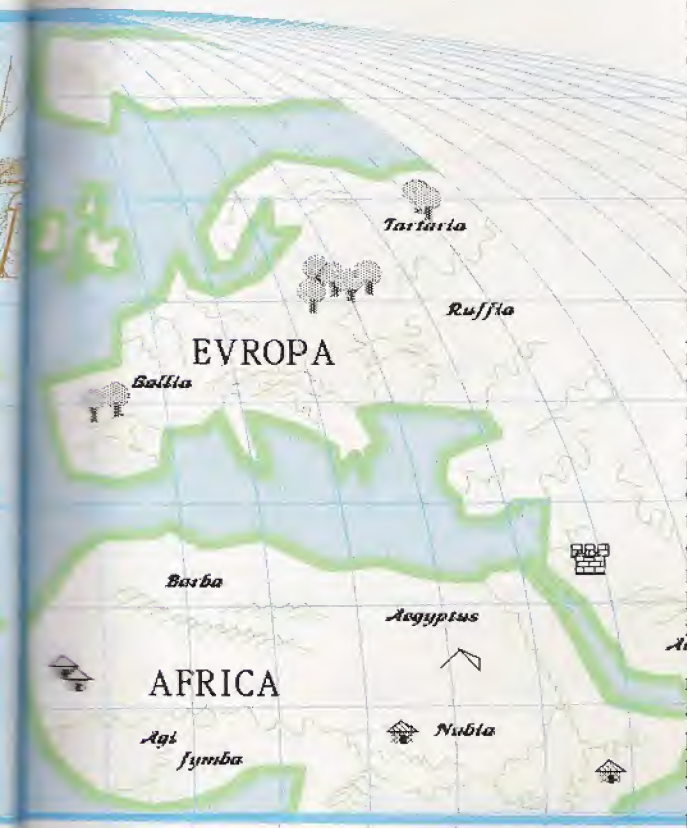
La dieta es un factor fundamental para determinar la fortaleza de un tripulante. Para mantenerse en forma, éste debe consumir los requerimientos semanales de fruta, verduras, carne y agua. Al comienzo del viaje, la fortaleza de cada miembro de la tripulación se establece en 100 en la matriz TS(,). El programa hace un cálculo semanal para comprobar si hay suficientes existencias de cada alimento para toda la travesía. De descender las existencias de algún tipo determinado, se le dice al jugador que la tripulación debe empezar a consumir media ración de ese alimento. Las medias raciones sólo se suministrarán durante una semana y reducirán en cinco unidades la fortaleza de la tripulación.

De acabarse algún tipo de alimento, la fortaleza de cada tripulante se reduce en 10 unidades por cada semana que pasen sin él. Por ejemplo, si la tripulación se quedara sin fruta y estuviera a media ración de agua, su tasa de fortaleza semanal se reduciría en 15. Si el agua se terminara por completo, la tasa de fortaleza se reduciría entonces en 20 por semana. Si, tras estar a media ración durante una semana, quedaran suficientes alimentos o agua para volver a las raciones completas durante el resto del viaje, se repartirán automáticamente al comienzo de la semana siguiente.

El estado de salud de la tripulación se da al comienzo de cada semana mediante la subrutina de la línea 4000. Hay cuatro estados de salud, determinados por la tasa de fortaleza. La clasificación más alta es *muy sano* (tasa de fortaleza 75-100), seguida de *sano* (50-75), *enfermo* (25-50) y *muy enfermo* (1-25). Cuando la tasa de fortaleza llega a 0, el tripulante muere y es enterrado en el mar, pero los salarios del fallecido se seguirán teniendo en cuenta cada semana y, cuando el barco llegue a puerto, se le abonarán al pariente más próximo. Esta subrutina también da la factura de los salarios para la próxima semana, los salarios totales del viaje hasta ese momento y el dinero que queda. El total de salarios acumulado se establece a cero al comienzo de esta sección, en la línea 800, y disminuye en función de la factura salarial semanal cada vez que se efectúa el bucle. Si la factura se vuelve mayor que el saldo de oro (comprobado por la subrutina de la línea 5000), se le advierte al jugador que se necesitará parte de los beneficios de la actividad comercial para pagar a la tripulación.

Si todo va bien el viaje durará ocho semanas, pero circunstancias imprevistas pueden aumentar este valor. La línea 801 crea una variable en serie, H$, para indicar si la tripulación se halla a medias raciones. La rutina determina la fortaleza y categoría de cada tripulante mediante la comprobación de la matriz de categoría/fortaleza, TS(,). Prepara un bucle en la línea 4060 para cada posible miembro. En la matriz hay espacio para información sobre 16 tripulantes, de modo que el bucle se establece en 16. Si la categoría de tripulación está registrada como 0, esa sección de la matriz estará vacía, de modo que la línea 4070 lleva al programa hasta la línea 4110. Si en la matriz hay registrado un tripulante, la línea 4075 imprime la descripción.

Cuando la salud de un tripulante llega a 0, la muerte debe registrarse en el diario de navegación del capitán. Puesto que una tasa de fortaleza de cero indica la muerte, cada sección vacía de la matriz parecería indicar un tripulante fallecido. Para evitar este problema, cuando una tasa alcanza el valor 0 se la restablece a −999 durante una semana. Al final de la semana se vuelve a establecer en 0 y al comienzo de la semana siguiente el programa informa sobre el fallecimiento. El restablecimiento en 0 hace que el programa ignore esa tasa durante las siguientes semanas. Ello evita que se imprima al



empezar cada semana un registro de los tripulantes fallecidos. Las líneas 4080-4098 verifican la fortaleza de los tripulantes restantes. Sus tasas se cotejan con los cuatro estados de salud, y la línea 4099 imprimirá el estado de cada uno. El bucle fortaleza-categoría termina en la línea 4110, que vuelve a enviar el programa a examinar la siguiente sección de la matriz.

La factura salarial semanal se calcula utilizando otro bucle entre las líneas 4120 y 4135 y se emplea una vez para cada una de las categorías de tripulantes. Los salarios para cada categoría se calculan mediante la ecuación de la línea 4130, multiplicando el contador para cada categoría por la tasa salarial, y acumulándolo en la variable WW. Creada y establecida en 0 al comienzo de cada semana mediante la línea 4119, WW representa el salario semanal, que se imprime en la línea 4145. Luego se suma al total para el viaje hasta ese momento (WT) y se imprime en las líneas 4160 y 4165. El jugador puede compararlo con el dinero que aún le queda, que se imprime en la línea 4175.

La subrutina de la línea 5100 maneja el reparto de raciones entre la tripulación. Utilizando un bucle FOR...NEXT para repetir el proceso para cada tipo de ración, se realizan varias comprobaciones del nivel de raciones restantes. La primera de ellas es una comprobación para asegurar que queden raciones. Esto se puede verificar simplemente comprobando que la cantidad de alimento, retenida en PA(), sea mayor que cero. De no ser éste el caso, la fortaleza de cada tripulante se reduciría en función de un factor de 10. Esta reducción la maneja realmente una pequeña subrutina en la línea 9300, que reduce la tasa de fortaleza para cada tripulante en función del valor de WF. Esta subrutina, por consiguiente, se puede utilizar para reducir la fortaleza de la tripulación en cantidades variables. Antes de llamar a la subrutina, sólo hemos de establecer WF en el valor deseado.

Si la cantidad de provisiones restantes es mayor que cero, entonces se realiza otra comprobación para ver si quedan provisiones suficientes para el resto del viaje. Esta cantidad se calcula en la línea 5180 a partir de la cantidad de tripulantes, CN, las necesidades semanales de alimentos, PN(), y el número de semanas que aún faltan para terminar la travesía. Si la cantidad de provisiones restantes es menor que las exigencias planificadas, se le da al jugador la opción de poner a la tripulación a media ración de ese tipo de alimento. El estado de cada tipo de alimento (ya sea ración completa o media ración) se indica mediante la matriz HR(), DIMensionada en la línea 802. Cuando las raciones son completas, el elemento de la matriz se establece en 1; si el alimento determinado se reduce a la mitad, entonces se establece en 0.5 el elemento correspondiente de la matriz. Si la tripulación se halla a media ración, entonces se emplea la subrutina de la línea 9300 para disminuir en cinco la tasa de fortaleza de cada miembro de la tripulación.

La comprobación final que efectúa esta subrutina antes de repartir la ración es ver si la cantidad de alimento que queda es menor que la cantidad a repartir. Si así fuera, sólo se repartiría la cantidad disponible y, para indicar que no queda ninguna provisión, el elemento correspondiente de la matriz de provisiones se establecería en −999.

Complementos al BASIC

Spectrum:
Realice las siguientes modificaciones:

```
 847 LET I$=INKEY$:IF I$="" THEN GO TO 847
4010 CLS
4190 LET I$=INKEY$:IF I$="" THEN GO TO 4190
4205 CLS
4295 LET I$=INKEY$:IF I$="" THEN GO TO 4295
4305 CLS
4398 LET I$=INKEY$:IF I$="" THEN GO TO 4398
5010 CLS
5080 LET I$=INKEY$:IF I$="" THEN GO TO 5080
5103 CLS
5220 INPUT I$:LET I$=I$(1 TO 1)
5298 LET I$=INKEY$:IF I$="" THEN GO TO 5298
```

BBC Micro:
Realice las siguientes modificaciones:

```
4010 CLS
4190 I$=GET$
4205 CLS
4295 I$=GET$
4305 CLS
4398 I$=GET$
5010 CLS
5298 I$=GET$
```

Módulo cuatro: El viaje

Longitud de la variable del viaje

```
40 JL=8:REM DURACION DEL VIAJE
```

Bucle principal del viaje

```
800 WT=0
801 H$="N": REM INDICADOR DE MEDIA RACION
802 DIM HR(4):HR(1)=1:HR(2)=1:HR(3)=1:HR(4)=1
820 FOR WK=1 TO JL : REM BUCLE PRINCIPAL DEL VIAJE
825 GOSUB 4000: REM INFORME ESTADO TRIPULACION
830 GOSUB 4200: REM INFORME PROVISIONES
835 GOSUB 4300: REM INFORME OTRAS MERCANCIAS
840 GOSUB 9200: PRINTCHR$(147)
842 PRINT:PRINT:PRINT
843 S$="SE CALCULA QUE EL VIAJE*":GOSUB 9100
844 PRINT"DURARA AUN OTRAS";JL-WK+1;"SEMANAS"
845 GOSUB 9200
846 PRINT:S$=K$:GOSUB 9100
847 GET I$:IF I$=""THEN 847
850 GOSUB 5000:REM COMPROBAR FACTURA SALARIAL
855 GOSUB 5100:REM REPARTIR RACIONES
889 NEXT WK
```

Informe sobre el estado de la tripulación

```
4000 REM INFORME SOBRE EL ESTADO DE LA TRIPULACION
4010 PRINTCHR$(147)
4020 S$="          DIARIO DE NAVEG. DEL CAPITAN*":GOSUB 9100
4025 S$="  ------------------------------  *":GOSUB 9100
4030 GOSUB 9200
4035 PRINT"AL EMPEZAR LA SEMANA";WK
4040 S$="EL ESTADO DE LA TRIPULACION ES:*":GOSUB 9100
4045 GOSUB 9200:PRINT
4055 PRINT
4060 FOR T=1 TO 16
4070 IF TS(T,1)=0 THEN 4110
4075 PRINT C$(TS(T,1));"      (";
4078 IF TS(T,2)=-999 THEN S$="MUERTO !!!!!!)*":GOTO 4099
4080 IF TS(T,2)>75 THEN S$="MUY SANO)*":GOTO 4099
4085 IF TS(T,2)>50 THEN S$="SANO)*":GOTO 4099
4095 IF TS(T,2)>25 THEN S$="ENFERMO ! )*":GOTO 4099
4098 S$="MUY ENFERMO !!)*"
4099 GOSUB 9100:GOSUB 9200
4110 NEXT T
4115 GOSUB 9200:PRINT
```

Factura salarial semanal

```
4119 WW=0
4120 FOR T=1 TO 5
4130 WW=WW+(CC(T)*WG(T))
4135 NEXT
4140 S$="FACTURA SALARIAL PARA LA SEMANA*":GOSUB 9100
4145 PRINT WW;"PIEZAS DE ORO"
4150 GOSUB 9200
4155 WT=WT+WW
4160 S$="TOTAL SALARIOS DEL VIAJE HASTA AHORA:*":GOSUB
     9100
4165 PRINT WT;"PIEZAS DE ORO"
4170 GOSUB 9200
4175 PRINT"DINERO RESTANTE=";MO;"PIEZAS DE ORO"
4180 PRINT:S$=K$:GOSUB 9100
4190 GET I$:IF I$="*"THEN 4190
4199 RETURN
```

Informe sobre provisiones

```
4200 REM INFORME SOBRE PROVISIONES
4205 PRINTCHR$(147)
4206 PRINT"AL EMPEZAR LA SEMANA";WK:GOSUB 9200
4210 S$="TE QUEDAN LAS SIGUIENTES*":GOSUB 9100
4215 S$="PROVISIONES:*":GOSUB 9100
4220 PRINT:GOSUB 9200
4225 FOR T=1 TO 4
4226 IF PA(T)=0 OR PA(T)=-999 THEN 4240
4230 PRINT PA(T);U$(T);"S DE ";P$(T)
4232 X=INT(PA(T)/(CN*PN(T)))
4235 PRINT"(SUFICIENTE PARA";X;" SEMANAS)"
4239 GOSUB 9200
4240 NEXT
4290 PRINT:S$=K$:GOSUB 9100
4295 GET I$=IF I$=""THEN 4295
4299 RETURN
```

Informe sobre otras mercancías

```
4300 REM INFORME OTRAS MERCANCIAS
4305 PRINTCHR$(147)
4306 PRINT"AL EMPEZAR LA SEMANA";WK:GOSUB 9200
4310 S$="TAMBIEN POSEES:*":GOSUB 9100
4320 PRINT:GOSUB 9200
4322 IF OA(1)=0 THEN 4332
4325 PRINT OA(1);:S$="FRASCOS DE MEDICINA*":GOSUB 9100
4330 GOSUB 9200
4332 IF OA(2)=0 THEN 4342
4335 PRINT OA(2);:S$="ARMAS*":GOSUB 9100
4340 GOSUB 9200
4342 IF OA(3)=0 THEN 4352
4345 PRINT OA(3);:S$="SACOS DE SAL*":GOSUB 9100
4350 GOSUB 9200
4352 IF OA(4)=0 THEN 4362
4355 PRINT OA(4);:S$="BALAS DE TELA*":GOSUB 9100
4360 GOSUB 9200
4362 IF OA(5)=0 THEN 4372
4365 PRINT OA(5);:S$="CUCHILLOS*":GOSUB 9100
4370 GOSUB 9200
4372 IF OA(6)=0THEN 4380
4375 PRINT OA(6);:S$="JOYAS*":GOSUB 9100
4380 GOSUB 9200:PRINT
4382 PRINT "TE QUEDAN";MO;:S$=" PIEZAS DE ORO*":GOSUB
     9100
4384 GOSUB 9200
4397 PRINT:S$=K$:GOSUB 9100
4398 GET I$:IF I$=""THEN 4398
4399 RETURN
```

Subrutina factura salarial

```
5000 REM COMPROBAR FACTURA SALARIAL
5005 IF MT>MO THEN 5010
5008 GOTO 5099
5010 PRINTCHR$(147)
5020 PRINT:PRINT:PRINT
5025 S$="ENTRE LA TRIPULACION CORRE EL RUMOR*":GOSUB
     9100
5030 S$="DE QUE NO TIENES SUFICIENTE*":GOSUB 9100
5035 S$="ORO PARA PAGARLES CUANDO TERMINE*":GOSUB
     9100
5040 S$="EL VIAJE.*":GOSUB 9100
5045 GOSUB 9200:PRINT
5050 S$="LOS ANIMOS SE ESTAN EXALTANDO !!*":GOSUB 9100
5055 GOSUB 9200:PRINT
5060 S$="ESPEREMOS QUE CONSIGAS*":GOSUB 9100

5065 S$="OBTENER ALGUN BENEFICIO!*:GOSUB 9100
5066 GOSUB 9200
5070 PRINT:S$=K$:GOSUB 9100
5080 GET I$:IF I$="" THEN 5080
5099 RETURN
```

Subrutina reparto de raciones

```
5100 REM REPARTIR RACIONES
5103 PRINTCHR$(147)
5105 S$="  REPARTIENDO LAS RACIONES*  ":GOSUB 9100
5106 S$="  ------------------------  *  ":GOSUB 9100
5107 GOSUB 9200:PRINT"SEMANA";WK:PRINT
5108 H$="N"
5110 FOR T=1 TO 4
5112 HR(T)=1
5115 IF PA(T)>0 THEN 5180
5120 PRINT"NO QUEDA ";P$(T);" !!!":GOSUB 9200
5130 S$="LA TRIPULACION SE ESTA DEBILITANDO !!*":GOSUB
     9100
5135 WK=10:GOSUB 9300
5139 GOTO 5290
5180 X=(PN(T)*CN)*(JL-WK+1)
5185 IF PA(T)<X THEN 5200
5190 GOTO 5270
5200 PRINT"QUEDA POCA ";P$(T)
5205 GOSUB 9200
5210 S$="QUIERES PONER A LA TRIPULACION A*":GOSUB 9100
5215 PRINT"MEDIA RACION DE ";P$(T)
5220 INPUT I$:I$=LEFT$(I$,1)
5221 IF I$<>"S" AND I$<>"N" THEN 5220:REM ERROR EN
     ENTRADA
5225 IF I$="N" THEN 5270
5230 HR(T)=.5:H$="S"
5240 WF=5:GOSUB 9300
5250 S$="LA TRIPULACION SE ESTA DEBILITANDO!*":GOSUB
     9100
5270 X=PN(T)*HR(T)*CN
5272 IF X>PA(T)THEN X=PA(T)
5275 PA(T)=PA(T)-X
5280 IF PA(T)=0 THEN PA(T)=-999
5285 PRINT X;U$(T);"S DE ";P$(T);" REPARTIDAS"
5290 PRINT:GOSUB 9200:NEXT
5295 PRINT:S$=K$:GOSUB 9100
5298 GET I$:IF I$="" THEN 5298
5299 RETURN
```

Reducción de la fortaleza de la tripulación

```
9300 REM REDUCIR FORTALEZA TRIPULACION EN FUNCION DE WF
9310 FOR S1=1 TO 16
9320 TS(S1,2)=TS(S1,2)-WF
9330 IF TS(S1,2)<1 THEN TS(S1,2)=-999
9340 NEXT
9399 RETURN
```

Gestor económico

El Sanyo MBC-550 representa una interesante opción en el mercado de ordenadores de 16 bits basados en el Intel 8088

A pesar de la rápida caída de los precios de los procesadores de 16 bits, la mayoría de los ordenadores basados en esta tecnología, con la excepción del Sinclair QL, siguen siendo muy caros. Las causas de este hecho son difíciles de comprender. Sinclair Research ha demostrado que es posible producir de forma rentable una máquina de 16 bits fijándole un precio razonable, y, sin embargo, la mayoría de los micros comparables tienen un precio muy superior al de ésta. La razón probablemente pueda atribuirse al tipo de cliente que existe en el mercado para tales máquinas. Éstos tienden a ser usuarios de gestión que, en opinión de los fabricantes, parecen estar en condiciones de pagar un poco más. Esta apreciación parece bastante acertada en el caso de los ordenadores basados en la serie de procesadores Intel 8088 y, probablemente, esté relacionada con la política de ventas del líder del mercado, IBM.

Pero todavía hay una parte del mercado que no está en condiciones de adquirir muchos de estos micros, aunque necesiten un ordenador con fines de gestión empresarial. Hasta hace muy poco, este sector del mercado estaba ocupado por ordenadores de oficina basados en el procesador Z80, equipados con unidades de disco que les permitían ejecutar CP/M. No obstante, actualmente muchos fabricantes están percatándose del potencial de este sector menoscabado del mercado de gestión y están volcándose en la producción de máquinas económicas basadas en el 8088.

El Sanyo MBC-550 es una de las primeras de éstas en salir al mercado. Aunque Sanyo no declare de forma manifiesta que sea compatible con el IBM PC, el MBC-550 ejecuta *de facto* el sistema de archivos en disco MS-DOS estándar. La máquina se vende con una única unidad de disco o bien en versión de unidades gemelas.

Al igual que la mayoría de las máquinas de oficina, el MBC-550 posee dos unidades separadas, el teclado y el ordenador principal. La carcasa de ambos es una combinación de plástico y metal con un acabado metálico plateado. El teclado está dividido en tres secciones. En el extremo izquierdo hay cinco teclas de función programables que, usadas junto con la tecla Shift, producen un máximo de 10 funciones que varían según la aplicación que se esté utilizando en cada momento. En BASIC, estas teclas se pueden emplear como entradas de palabras clave individuales, con las instrucciones utilizadas más comúnmente, como LIST, LOAD y RUN.

El teclado de máquina de escribir tiene un buen trazado, con todas las teclas de control en sus posiciones habituales. Las teclas de control incluyen un Backspace con borrado y una tecla Insert/Delete, siendo operativa una sola de las dos en cada aplicación. Hay, asimismo, una gran tecla Return, cuyo tamaño es cuatro veces el de las teclas normales. A la derecha de la barra espaciadora hay una tecla Graphics que, al trabarse, produce en la pantalla caracteres de gráficos. En el extremo derecho del teclado hay un teclado numérico que se puede utilizar como calculadora o, de pulsarse la tecla Number Lock, como control del cursor, para posibilitar la edición completa en pantalla en el BASIC.

El ordenador propiamente dicho es una gran caja plana del tamaño de un aparato de video, que se ha diseñado del tamaño necesario para colocar sobre ella la pantalla opcional Sanyo. Al igual que el teclado, está construido en su mayor parte con láminas metálicas. En la parte frontal del ordenador está el interruptor de potencia y hay espacio para dos unidades de disco, si bien en nuestra fotografía vemos el modelo de una sola unidad. Las unidades que emplea el MBC-550 son del tipo estándar de discos flexibles de 5 ¼ pulgadas. En la parte delantera de cada unidad hay un clip para mantener el disco en posición y evitar que pueda ser sacado mientras la cabeza de la unidad lo está leyendo.

Diferencias fraternales
El Sanyo MBC-550 es una máquina de oficina de 16 bits que se vende a un buen precio y opera bajo el popular sistema operativo MS-DOS. La máquina viene en dos versiones: la MBC-550 es la versión de una sola unidad de disco; la versión que configura la doble unidad es la denominada MBC-555. La pantalla que vemos en la fotografía no está incluida en el precio básico

Chris Stevens

Interfaces para periféricos

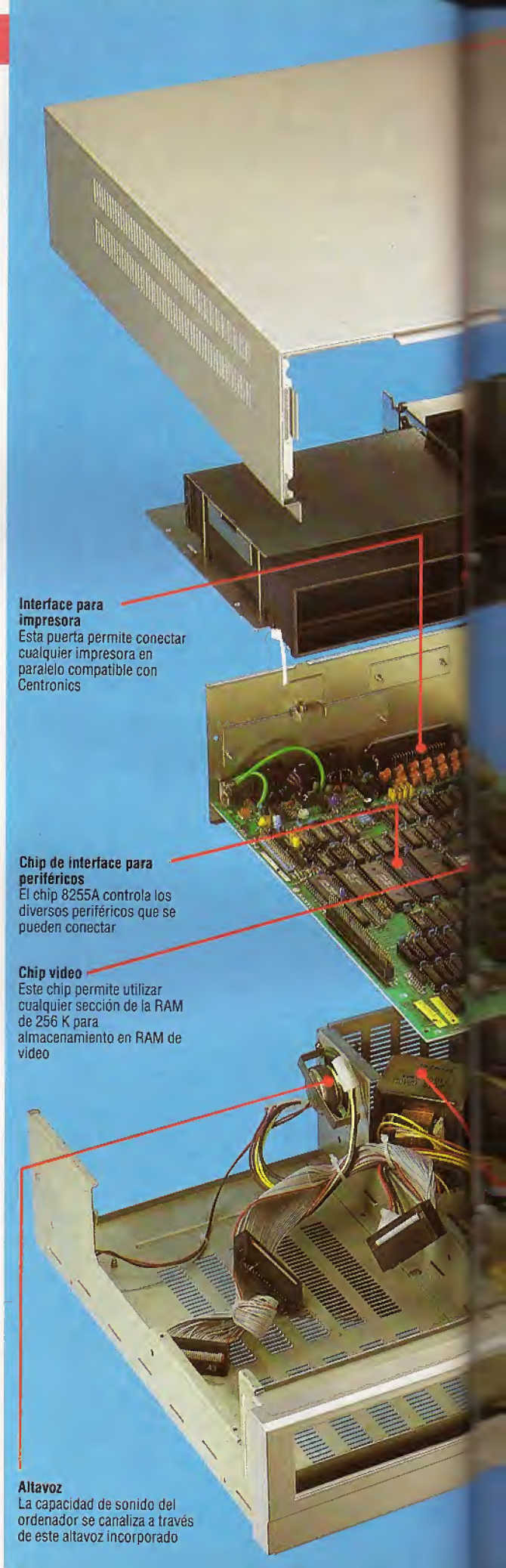
En su parte posterior, el ordenador contiene la fuente de alimentación eléctrica y las interfaces para periféricos. En el lado izquierdo de su parte

posterior se halla el cable de potencia, la caja del fusible y la conexión a tierra. A la derecha hay una interface en paralelo Centronics que permite la conexión de una impresora. A la derecha de ésta hay un par de conectores para pantalla. El primero es un conector RGB que permite que el ordenador haga funcionar una pantalla en color, mientras que el otro es un enchufe de video compuesto para la pantalla monocromática verde de Sanyo. Encima de estas interfaces hay otras puertas de ampliación que en el futuro darán cabida a interfaces extras para periféricos.

La puerta Line de encima de la interface para impresora se incluye para la conexión de una interface en serie RS232 que facilitaría la comunicación del ordenador con otras máquinas, a través de un modem, o con cualquier otro dispositivo en serie, como una impresora. Junto a ella hay una puerta en la que se puede instalar una interface para palanca de mando de la serie MBC o bien compatible con Atari. Ésta no sólo permite la instalación de una palanca de mando sino que también permitiría controlar el ordenador mediante un mando de bola, un mando de raqueta (*paddle*) u otro dispositivo externo. Hay, asimismo, un conector para bus externo que permite la conexión en interface de otros dispositivos periféricos.

A diferencia de muchos fabricantes que se oponen taxativamente a que los usuarios manipulen sus máquinas, en el manual para el usuario Sanyo ha incluido esmeradamente instrucciones para la instalación de todas estas interfaces para periféricos, bancos extras de RAM y una segunda unidad de disco por si hubiera instalada sólo una. Esta inclusión es muy de agradecer, aunque la guía menciona que, en caso de alguna duda, los usuarios deberían consultar con un ingeniero cualificado. El resto del manual para el usuario es igualmente valioso. Contiene un glosario de términos de informática, una descripción completa del BASIC Sanyo, incluyendo instrucciones para utilizar el sistema de disco desde BASIC, y gran cantidad de información técnica.

Espacio para ampliaciones
Aunque el MBC-550 no esté equipado con tantas opciones para periféricos como algunas de las máquinas más caras que existen en el mercado, Sanyo la ha diseñado para que sea ampliable. Hay ranuras extras para interfaces para palancas de mando y dispositivos en serie y en el manual para el usuario se ofrecen instrucciones para instalarlas

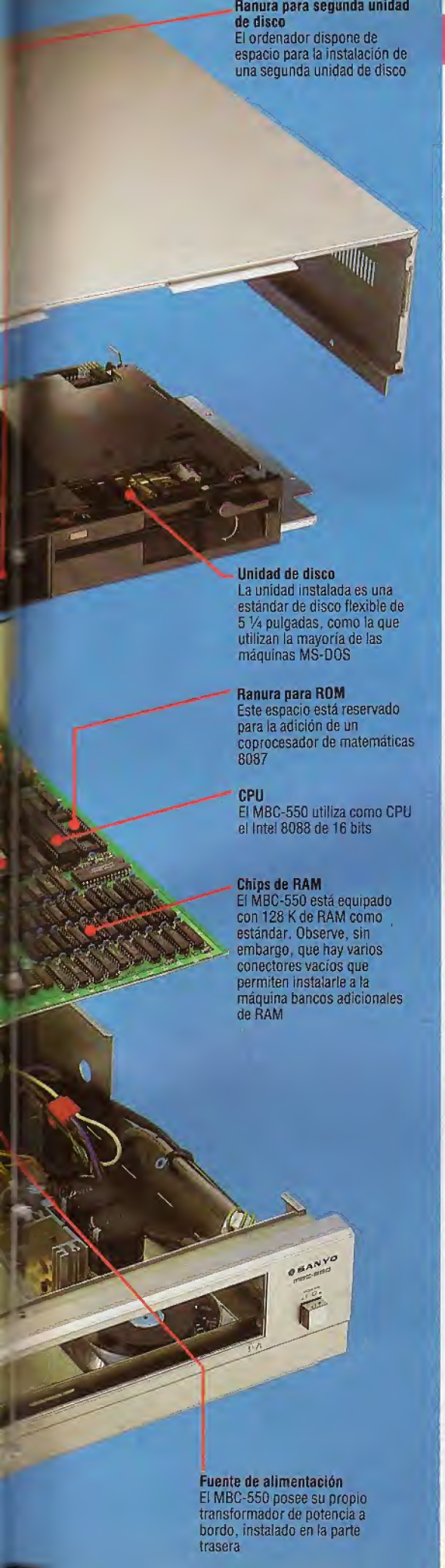


Interface para impresora
Esta puerta permite conectar cualquier impresora en paralelo compatible con Centronics

Chip de interface para periféricos
El chip 8255A controla los diversos periféricos que se pueden conectar

Chip video
Este chip permite utilizar cualquier sección de la RAM de 256 K para almacenamiento en RAM de video

Altavoz
La capacidad de sonido del ordenador se canaliza a través de este altavoz incorporado



El BASIC Sanyo, un derivado de la versión de Microsoft, es ciertamente adecuado, aunque parece ser algo lento para procesar funciones aritméticas. Ya se ha notado desde hace mucho tiempo que el chip 8088 es perezoso para procesar matemáticas (a ello se debe la popularidad del coprocesador de matemáticas 8087), pero el MBC parèce aún más lento que la mayoría de las otras máquinas basadas en el 8088. Del mismo modo, el dibujo de líneas en el MBC-550 es también muy lento. Esto se debe a que el BASIC Sanyo carece de una instrucción DRAW, lo que exige llevar a cabo el trazado de líneas mediante la instrucción PSET, que simplemente enciende un pixel especificado con un color dado. Por lo tanto, el trazado de una línea supone utilizar un bucle.

Aparte de instrucciones estándares tipo Microsoft, tales como MID$, LLIST y CIRCLE, también hay instrucciones para definir y contemplar ventanas en la pantalla. Se fomenta la programación estructurada mediante la inclusión de la sentencia condicional WHILE...WEND. Muchas palabras clave del BASIC Sanyo se pueden entrar mediante dos o tres pulsaciones de tecla basadas en la tecla Control. Por ejemplo, la palabra clave DIM se puede entrar pulsando simultáneamente CTRL, SHIFT y D, mientras que PRINT se entra pulsando CTRL y P. Si bien todo intento por simplificar la entrada de palabras clave del BASIC es digno de elogio, y muchas de las instrucciones con pulsaciones de tecla abreviadas guardan alguna relación con los originales, es difícil imaginarse que alguien intente siquiera recordar las 40 instrucciones disponibles para escribir programas en BASIC. Lo más probable es que intente memorizar algunas de las versiones abreviadas empleadas más comúnmente.

Con el Sanyo MBC-550 viene un disco de sistema MS-DOS, que contiene también el BASIC Sanyo, el paquete para tratamiento de textos *WordStar*, tan ampliamente utilizado, y la hoja electrónica *CalcStar*. Estos paquetes van acompañados de un manual que ofrece descripciones completas para su empleo.

Cabría esperar que el MBC-550 estándar (basado en el chip 8088 y operando bajo MS-DOS) tuviera pocos problemas para ejecutar la ingente cantidad de software IBM PC que existe en el mercado. Pero lo cierto es que las restricciones de hardware de la máquina significan que ninguno de los paquetes compatibles con IBM que probamos en el equipo se pudo ejecutar.

Sin ninguna duda, existe un mercado para máquinas de oficina MS-DOS de precio reducido capaces de ejecutar software compatible con IBM. No se trata ya de esperar que aparezcan, sino de cuándo aparecerán. El Sanyo MBC-550 es un primer intento, pero, lamentablemente, su falta de compatibilidad lo dejará en una situación de aislamiento.

Para quien sea propietario de una pequeña empresa que sólo necesita un ordenador para ejecutar facilidades competentes de tratamiento de textos y hoja electrónica, el MBC-550 parece una buena propuesta.

No obstante, quien desee tener acceso a una base de software más amplia de la que hay disponible para el Sanyo MBC-550, deberá resignarse ya sea a gastar algo más o bien a esperar un poco más de tiempo.

SANYO MBC-550

DIMENSIONES
375×355×108 mm

CPU
Intel 8088 operando a 3,6 MHz

PANTALLA
Pantalla para texto de 80×25, dando 640×200 pixels en modalidad de alta resolución. En alta resolución hay hasta ocho colores disponibles

INTERFACES
Puerta en paralelo Centronics, con opciones para la instalación de una interface RS232, y una puerta para palanca de mando compatible con Atari

LENGUAJES DISPONIBLES
BASIC Sanyo

TECLADO
65 teclas tipo máquina de escribir, cinco teclas de función doble programables y un teclado numérico de 19 teclas

DOCUMENTACION
El manual es muy bueno y contiene una explicación completa de las palabras clave del BASIC, aunque, lamentablemente, no contiene un curso de aprendizaje. Asimismo, hay una introducción al MS-DOS, *WordStar* y *CalcStar*. Insólitamente, tratándose de una máquina de oficina, el manual para el usuario proporciona gran cantidad de información técnica, incluyendo instrucciones que le indican al usuario cómo instalar sus propias placas de ampliación

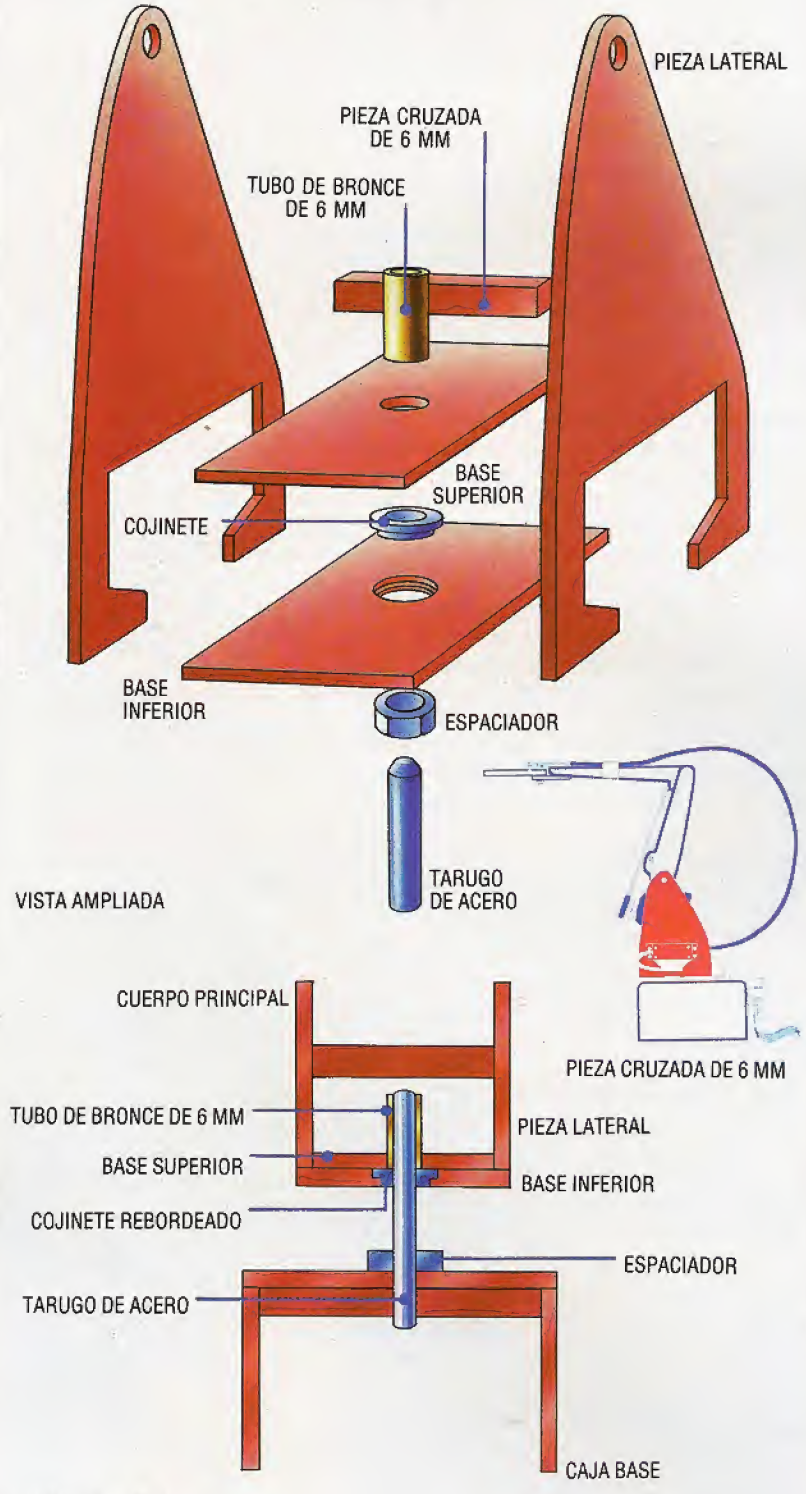
VENTAJAS
Por su precio, el Sanyo MBC-550 es, ciertamente, una buena compra, proporcionando una informática de gestión completa de 16 bits a una fracción del precio normal

DESVENTAJAS
Debido a que la máquina carece de compatibilidad con la mayor parte del software existente, el ordenador se encuentra con el problema habitual de las máquinas nuevas: la falta de programas disponibles

Ensamblaje del brazo

Ensamblaremos el cuerpo principal, las secciones superior e inferior del brazo y efectuaremos las junturas del codo y el hombro

Paso 1: Cuerpo principal



Paso 1: Cuerpo principal

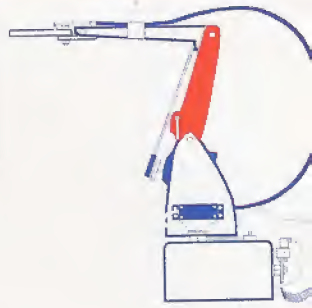
Taladre un agujero en la pieza inferior de la base (la más ancha de las dos piezas) para aceptar el cojinete. Utilizando una broca de mayor tamaño o un avellanador, bisele el labio superior del agujero para asentar en él el reborde del cojinete, como se ve en la ilustración. Después taladre un agujero de 6 mm de diámetro en la pieza de la base superior para recibir el trozo de 25 mm de tubo de 6 mm de diámetro interno. Con los agujeros alineados y el cojinete asentado sobre la pieza inferior de la base, pegue las dos piezas entre sí. Inserte el tubo de bronce en el agujero, desde arriba, y péguelo en su sitio. Cuando este ensamblaje esté seco se pueden añadir las piezas laterales del cuerpo. Perfore los agujeros en la parte de arriba de las piezas laterales de modo que acepten estrechamente un trozo de tubo de 4 mm de diámetro externo: se trata de un ajuste de fricción. Inserte un trozo de madera de 6×6 mm y 37 mm de largo a modo de pieza cruzada entre los dos laterales (colocada en la parte trasera del ensamblaje, en línea con el corte para el motor de la pieza lateral) y pegue el ensamblaje. Cuando éste esté seco, coloque un espaciador (p. ej., una tuerca grande) en el tarugo de acero que sobresale desde la parte superior de la caja base ensamblada anteriormente. Empuje el tarugo a través del cojinete y compruebe que el ensamblaje del cuerpo principal pueda girar libremente.

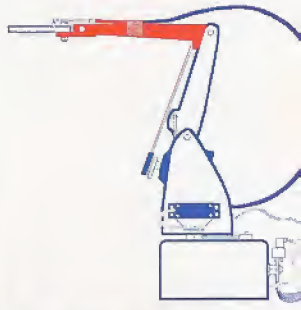
Paso 2: Brazo inferior

Corte un trozo de 37 mm de tubo de bronce de 5 mm de diámetro externo y taladre agujeros en el medio de las piezas del brazo inferior para recibir este tubo. Asegúrese de obtener un ajuste apretado, puesto que este tubo de bronce formará parte de la juntura del hombro. Perfore otros dos agujeros en el extremo piramidal de las piezas del brazo inferior para recibir el tubo de bronce de 4 mm de diámetro externo. Como antes, asegúrese de conseguir un ajuste de fricción. Trabajando en el medio, pase el tubo de 5 mm a través de los agujeros de las piezas del brazo inferior utilizando, como vemos en la ilustración, uno de los servomotores como espaciador. Corte dos trozos de 20 mm de largo de madera de 6 mm como piezas cruzadas y colóquelas entre las piezas del brazo inferior. Éstas van justo debajo del tubo de bronce y en la parte de abajo del ensamblaje, y se emplean para montar el servomotor. Péguelos en su sitio y, una vez secos, monte el motor utilizando cuatro tornillos sin tuerca y arandelas de goma. Ya se puede unir el ensamblaje del brazo inferior al cuerpo principal, por la juntura del hombro. Ahora inserte un trozo de 45 mm de largo de tubo de 4 mm a través de uno de los agujeros de la parte superior del cuerpo principal y a través del tubo de bronce que ya estaba colocado en el ensamblaje del brazo inferior. Asegúrese de que el ensamblaje del brazo inferior esté centrado. Marque la posición en la que parezca estar correctamente alineado, desmóntelo, y luego pegue el tubo de bronce más grande en su lugar en el ensamblaje del brazo inferior. Una vez seco, vuelva a ensamblar la juntura del hombro.

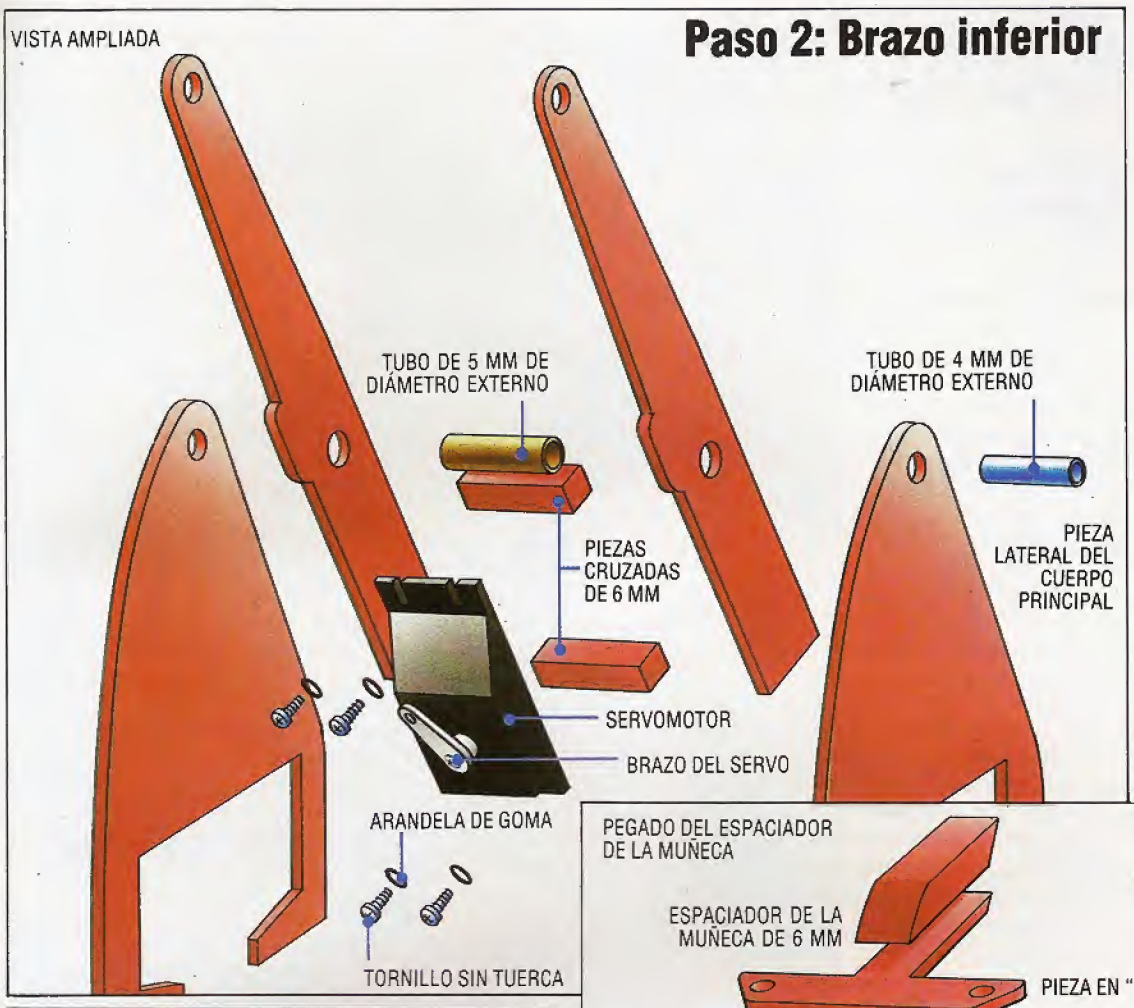
Paso 3: Brazo superior

Perfore dos agujeros para recibir el tubo de bronce de 5 mm en los extremos más anchos de las piezas del brazo superior. Perfore otros dos agujeros en los otros extremos. Pegue un trozo de 25 mm de madera de 6×6 mm en la pieza en "T" y moldee los extremos. Éste actuará a modo de espaciador de la muñeca. Coloque la sección en "T" entre las dos piezas del brazo superior y taladre a través del espaciador de la muñeca de modo que a través del espaciador y de las secciones del brazo superiores se pueda pasar un tornillo para metales. Fije la pieza en "T" en su sitio con una arandela y un tornillo. En el otro extremo del brazo, empuje a través de los agujeros taladrados un trozo de 17 mm de tubo de bronce de 5 mm, y conecte el ensamblaje del brazo superior al brazo inferior en la juntura del codo pasando a través de la juntura, como antes, un trozo de 25 mm de tubo de 4 mm. Compruebe que el brazo superior esté centrado antes de desmontarlo y pegar en su sitio el tubo de bronce de diámetro mayor.

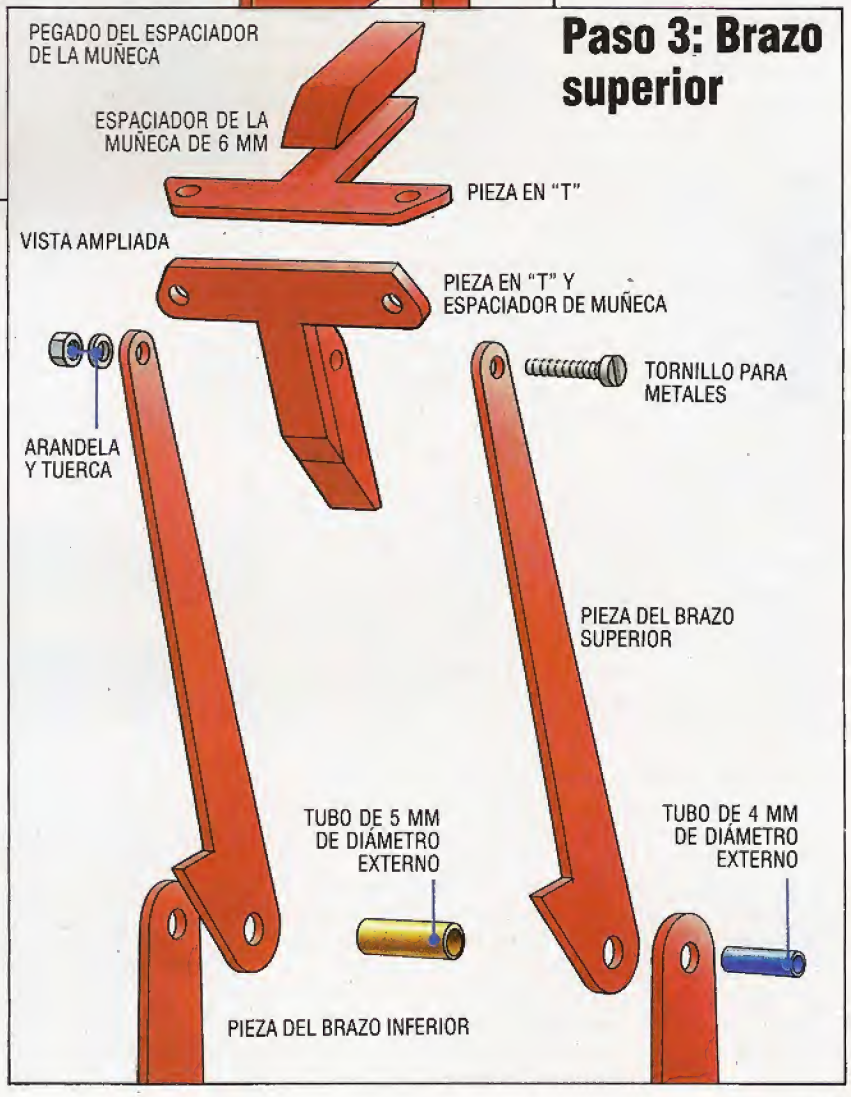


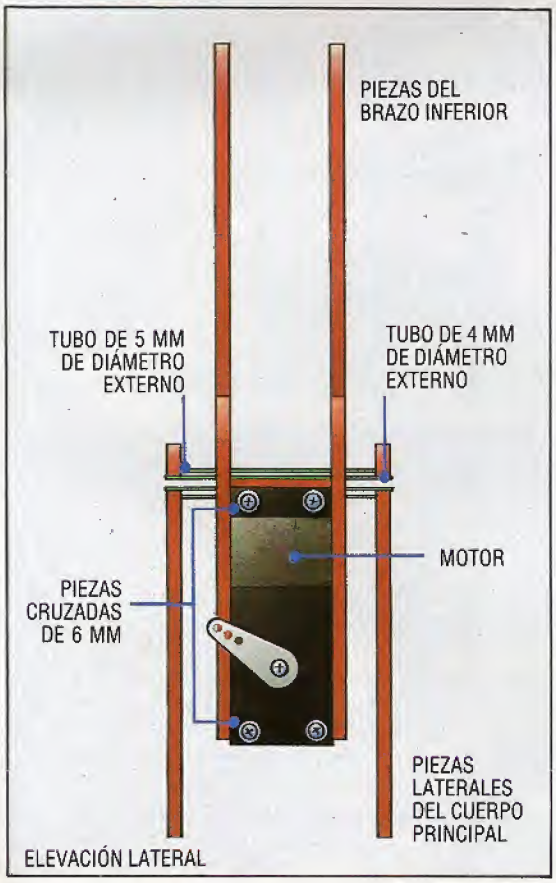


Paso 2: Brazo inferior



Paso 3: Brazo superior





Kevin Jones

Conjuntos

Veamos cómo se utilizan los conjuntos en PASCAL y cuál es el orden de precedencia de los operadores de conjuntos

Además de sus estructuras de datos típicas, tales como matrices y archivos, el PASCAL incluye conjuntos y registros. Examinemos los primeros.

Con frecuencia hablamos del *juego* o *conjunto* de caracteres de un ordenador. Pero, ¿qué es precisamente un *conjunto* y en qué se diferencia de una matriz? La principal característica de un conjunto reside en que es un grupo de objetos que se pueden procesar como una única entidad, en vez de tener que acceder a cada elemento de forma individual. Un ejemplo práctico sería el conjunto de todas las personas que programan en PASCAL, o el de las letras del alfabeto que son vocales. Con frecuencia no hay implicado ningún orden particular, siendo la única pregunta de interés: ¿es o no es este objeto determinado un miembro del conjunto?

Por razones de eficacia de implementación, el PASCAL impone algunas restricciones a los conjuntos. Sólo pueden tener tipos escalares simples como miembros (no matrices ni otros datos estructurados) y hay un límite máximo (definido por implementación) para la gama permitida para este tipo "base". La sintaxis es directa; por ejemplo:

```
TYPE
  Numeros    =SET OF 0..127;
  Alfabeto   =SET OF 'A'..'Z';
  MezclaCol  =SET OF (Rojo,Verde,Azul);
```

La gama de los valores posibles del tipo base ordinal del conjunto se puede expresar con la misma sintaxis que para tipos de subrango.

Las variables de conjunto se declaran en la parte de declaración VAR, de la misma forma que se declaran todas las variables en PASCAL, y pueden aparecer en sentencias literalmente (es decir, como "literales") encerradas entre corchetes:

```
VAR
  codigos   : Numeros;
  paleta    : MezclaCol;
BEGIN
  codigos   :=[0..2,4,8,16,32,64];
  paleta    :=[Rojo..Azul];{etc.}
```

Este segmento inicializaría los códigos de conjunto de modo que contuvieran sólo los números del 0 al 2 inclusive y 4,8, etc., tal como se listan. Dado que en el conjunto no hay ningún orden inherente (al contrario que en su tipo base), podrían igualmente expresar este conjunto como [64,32,16,8,4,0..2], pero observe que el subrango 2..0 (ilegal en una verdadera definición de subrango) indicaría meramente un rango vacío. De hecho, se puede inicializar como conjunto vacío cualquier conjunto, mediante la sentencia: Conjunto1:=[], que da origen a la única excepción a la regla general del PASCAL, a saber: el tipo de cualquier literal se conoce mediante la inspección. Sin que al menos un miembro del conjunto aparezca de forma literal, ni nosotros ni el ordenador podemos determinar su tipo. Afortunadamente, el conjunto vacío sólo puede ocurrir en asignaciones (como en el ejemplo) o en expresiones en las cuales los otros identificadores ya tendrán declarados sus tipos. Ello significa, sin embargo, que el conjunto vacío es un subconjunto de todos los tipos de conjuntos, pero eso es natural.

Uno de los operadores más útiles que proporciona el PASCAL para estructuras de conjuntos es, como DIV y MOD, una palabra reservada: IN. Nos permite comprobar la pertenencia a un conjunto y es un operador de relación que toma dos operandos. El lado izquierdo debe ser una expresión que evalúe uno de los posibles miembros del conjunto (en otras palabras, un valor del tipo base del conjunto) y el lado derecho puede ser una variable de conjunto o un literal de conjunto. La expresión dará un resultado booleano: verdadero si el valor pertenece al conjunto y falso en caso contrario.

Por consiguiente: N IN codigos y Verde IN paleta son expresiones booleanas legales. La comprobación habitual de si un carácter es un dígito se podría escribir así:

```
IF(c>='0')AND(c<='9')THEN...
```

Mucho menos confuso sería comprobar si el valor de c pertenece al conjunto de caracteres en el cual estamos interesados, utilizando:

```
IF c IN('0'..'9')THEN...
```

O, quizá, si estamos escribiendo un programa para un juego de naipes:

```
TYPE
  rango=(dos,tres,cuatro,cinco,seis,siete,ocho,
         nueve,diez,Valet,Dama,Rey,As);
  ConjNaipes=SET OF rango;
VAR
  naipe : rango;
  figuras : ConjNaipes;
BEGIN
  figuras:=(Valet..As);
  IF naipe IN figuras THEN {etc.}
```

Compare este último ejemplo con:

```
IF(naipe=Valet)OR(naipe=Dama)OR(naipe=Rey)
OR...
```

En PASCAL también hay operaciones definidas sobre conjuntos como un todo. Éstas son intersección, unión y diferencia. Si B es el conjunto de todos los programadores de BASIC y P representa a los de PASCAL, la intersección de ambos conjuntos es el conjunto de programadores que utilizan el BASIC y el PASCAL *a la vez*. La unión de P y B es el conjunto de personas que programan o en PASCAL o en BASIC, es decir, la *combinación* de los dos conjuntos.

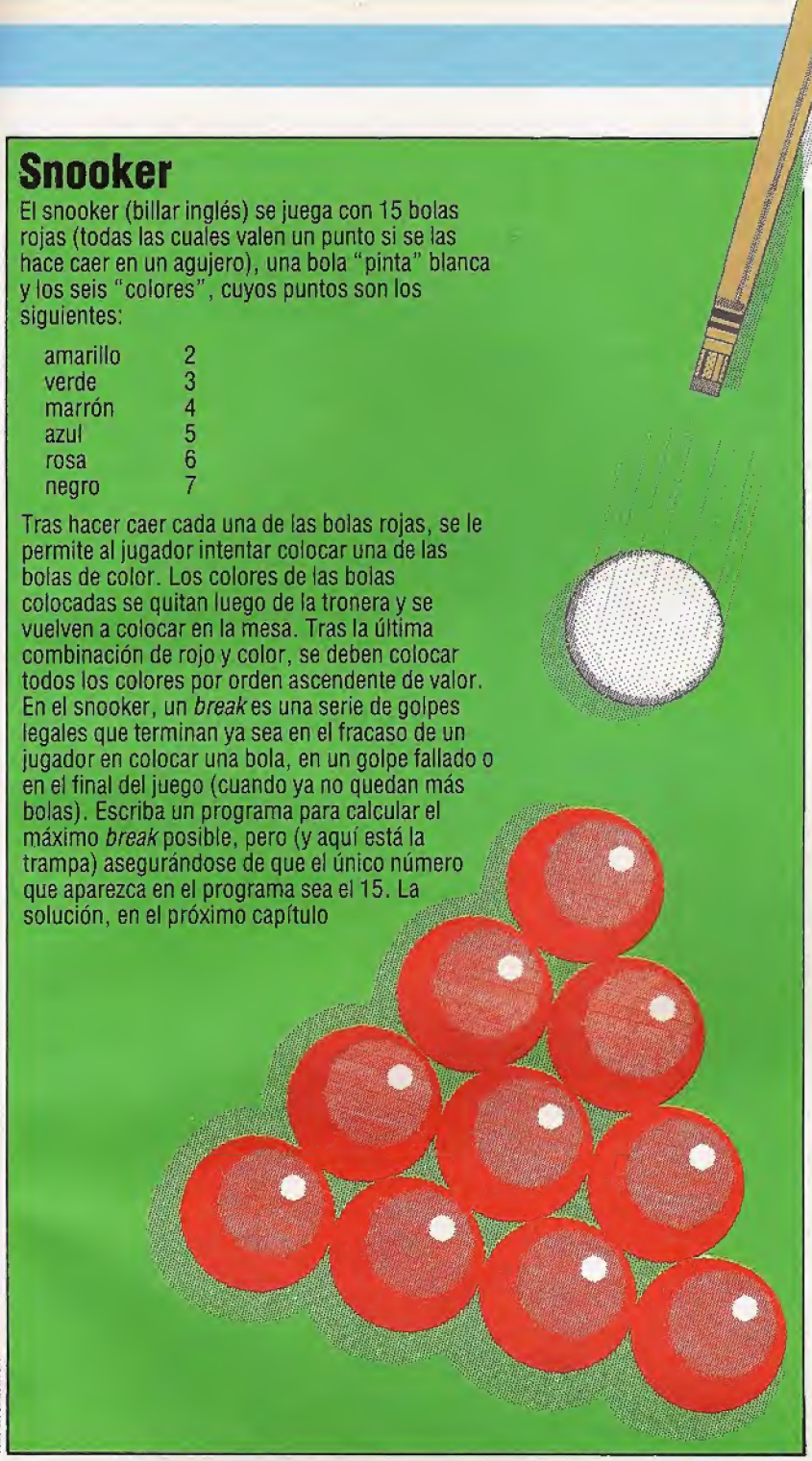
La diferencia de conjuntos es, como su nombre sugiere, el resultado de suprimir o restar todos los miembros de un conjunto del otro. Por consiguiente, en la notación del PASCAL, P−B representa a

Snooker

El snooker (billar inglés) se juega con 15 bolas rojas (todas las cuales valen un punto si se las hace caer en un agujero), una bola "pinta" blanca y los seis "colores", cuyos puntos son los siguientes:

amarillo	2
verde	3
marrón	4
azul	5
rosa	6
negro	7

Tras hacer caer cada una de las bolas rojas, se le permite al jugador intentar colocar una de las bolas de color. Los colores de las bolas colocadas se quitan luego de la tronera y se vuelven a colocar en la mesa. Tras la última combinación de rojo y color, se deben colocar todos los colores por orden ascendente de valor. En el snooker, un *break* es una serie de golpes legales que terminan ya sea en el fracaso de un jugador en colocar una bola, en un golpe fallado o en el final del juego (cuando ya no quedan más bolas). Escriba un programa para calcular el máximo *break* posible, pero (y aquí está la trampa) asegurándose de que el único número que aparezca en el programa sea el 15. La solución, en el próximo capítulo



Ian McKinnell

Precedencia de operadores

En la tabla de abajo vemos el orden de precedencia de los operadores. El paréntesis obliga a evaluar por separado la expresión encerrada, rompiendo el orden normal de precedencia (que se refleja aquí) si fuera necesario

Precedencia	Operadores	Tipo
Máxima	NOT + -	{unarios}
	AND */ DIV MOD	{multiplicación}
	OR +-	{adición}
Mínima	<<=<>>=>IN	{de relación}

todas las personas que programan en PASCAL pero no en BASIC. Del mismo modo, la notación que se emplea para uniones es P+B, y para la intersección P*B. Estos símbolos de operadores resultan ser los mismos que los bien conocidos operadores aritméticos, pero no se debe confundir la distinta clase de operaciones. En realidad, la razón por la que aparecen con tanta naturalidad en el contexto de las operaciones con conjuntos, es porque representan la comprobación de bits involucrada.

Tomemos un conjunto de ocho elementos. La presencia de un miembro determinado se podría indicar estableciendo el bit apropiado de un patrón de 8 bits (un byte); la ausencia en el conjunto se podría señalar, del mismo modo, mediante un cero. Las comprobaciones de pertenencia, entonces, requieren apenas una máscara o comprobación de bit, una unión de conjuntos se convierte en una operación OR y la intersección en un simple AND. Estas operaciones están invariablemente disponibles a nivel de lenguaje máquina, de modo que el compilador de PASCAL puede proporcionar estructuras de datos de conjuntos y las operaciones con ellas relacionadas con gran eficacia. El gasto de memoria es asimismo mínimo y, especialmente cuando el tamaño de los conjuntos se puede mantener dentro del tamaño de palabra del ordenador (en máquinas de 32 bits y más, por ejemplo), las operaciones con conjuntos pueden ser las que se implementan con mayor eficiencia en PASCAL.

Estamos ahora en condiciones de resumir todos los operadores del PASCAL. Aquellos de los que no hemos hablado de forma explícita son ya familiares en otros lenguajes, y el PASCAL facilita mucho las cosas al tener sólo cuatro niveles de precedencia de operadores. Naturalmente, los operadores *unarios* o *monádicos* son los que tienen precedencia sobre todos los demás. Éstos son los símbolos + y −, que indican el signo de un número, y el operador de negación booleana NOT. En el segundo nivel de precedencia se hallan todos los operadores de "multiplicación" (incluyendo los signos de división) seguidos por la suma/resta y, en el nivel inferior de precedencia, todos los operadores de relación, incluyendo IN.

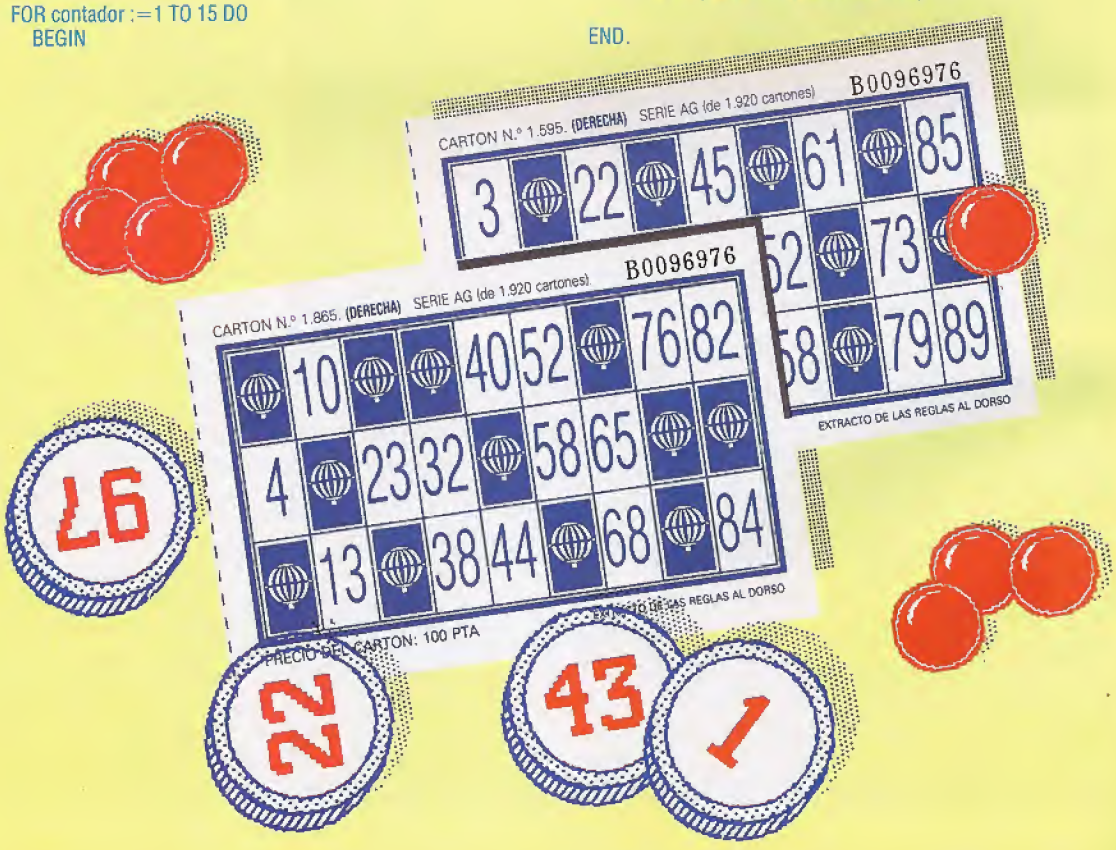
Observe que los otros dos operadores booleanos (AND y OR) se tratan de forma correcta como operadores de multiplicación y adición, respectivamente. Ello evoca de manera fidedigna la verdadera álgebra booleana implicada, y significa que muchas condiciones de relación se encierran entre paréntesis para anular esta precedencia. De lo contrario, por ejemplo: IF N>0 AND N<10 THEN.. dará un error en tiempo de compilación, porque la expresión 0 AND N (que se evaluaría primero) intenta combinar dos operandos enteros con un operador booleano.

Los operadores del mismo nivel de precedencia se evalúan de izquierda a derecha, como es habitual. El símbolo utilizado para la asignación (:=) tiene una precedencia inferior a cualquiera de los operadores anteriores, porque la expresión del lado derecho de una asignación debe haber sido evaluada completamente antes de efectuar la asignación. Un último aviso: uno jamás puede dar por sentado que no se evaluará alguna parte de una expresión, de modo que: IF(N>0)AND(K/N<10) THEN... podría colgar el programa si N fuera cero (¡lo que crearía un error de división por cero!).

¡Bingo!

Un cartón de bingo se puede representar mediante un conjunto. Aunque los elementos base (enteros comprendidos entre el 1 y el 90) están ordenados, la única consideración vital es si el número cantado está o no en el cartón. Al decir del PASCAL: "numero IN Carton" es o bien verdadero, o falso. El programa simula una partida de bingo leyendo primero los números del cartón desde el teclado y luego comprobando si el conjunto de los números cantados es un subconjunto del cartón. La expresión Carton-Cantado se convertirá en el conjunto vacío cuando todos los miembros de Carton estén también en el conjunto denominado Cantado. La adición de un miembro a un conjunto que ya lo contenga no provoca ninguna modificación en el conjunto, así como "quitar" un no miembro. Observe que no podemos añadir directamente un miembro a un conjunto; pero creado un conjunto de un solo elemento, encerrando el número entre corchetes, podemos obtener la unión de dos conjuntos. Tal como está, el programa permitirá entradas de cartón duplicadas, y aceptará un número ilegal fuera de la escala 1-90, produciendo un error de ejecución. A modo de ejercicio, piense en alguna forma sencilla de añadir construcciones de bucle para evitar estas anomalías y rechazar los números ya cantados

```
PROGRAM          Bingo     (input,output);

CONST
          Columnas      = 40;     {segun medida VDU}
          Medio         = 25;     {segun medida VDU}

TYPE
          Bingo         = SET OF 1..90;

VAR
          contador      : 1..15;
          Reconocido,
          Cantado,
          Vacio,
          Carton        : Bingo;
          numero        : integer;
          Casa          : boolean;

BEGIN
  Vacio :={ };
  Reconocido :=[1..90];
  WriteLn ('***BINGO***' : Medio);
  WriteLn;
  WriteLn ('Entre los 15 numeros del carton,' );
  WriteLn ('(un RETURN tras cada uno de ellos) :');
  Carton := Vacio;

FOR contador :=1 TO 15 DO
  BEGIN
    write (contador : 10, ': ? ');
    ReadLn (numero);
    Carton :=Carton+[numero]
  END;

  WriteLn;
  WriteLn ('PREPARADO !' : Medio);
  WriteLn;
  WriteLn ('Ahora cante cada numero :');
  Cantado := Vacio;

  REPEAT
    write ('?': Medio);
    ReadLn (numero);
    Cantado :=Cantado+[numero];
    Casa := Carton - Cantado = Vacio

  UNTIL Casa;

  WriteLn;
  WriteLn ('Felicitaciones !' : Medio);
  WriteLn ('Sus numeros fueron:');
  WriteLn;

  FOR numero :=1 TO 90 DO
    IF numero IN Carton THEN
      write (numero : Columnas DIV 8)

END.
```



Última llamada

Examinadas ya las rutinas más importantes que componen el OS del BBC Micro, concluiremos el estudio con algunos programas que ilustran los usos de las llamadas del OS y dejan en evidencia algunas de sus limitaciones

En el capítulo anterior dimos un programa como ejercicio que muestra muy bien cómo podemos interceptar uno de los vectores del OS para cambiar el modo como responde éste a ciertos eventos. El programa altera el contenido de OSWRCHV, el vector que contiene la dirección de la rutina OSWRCHV. Este vector se encuentra en las direcciones &20E y &20F.

Una de las funciones de OSWRCH es tomar el texto en BASIC e imprimirlo en la pantalla cuando se lista un programa. La principal función del código máquina en nuestro programa es examinar el código ASCII de cada carácter según pasa por el acumulador durante la rutina OSWRCH. Si el código ASCII está entre el 64 y el 91 (correspondiente a las mayúsculas), entonces se agrega 32 al valor para dar su equivalente en minúsculas. De esta manera el programa forma una "cuña" que se ejecutará cada vez que el sistema operativo llame a OSWRCH. Después de que se ha ejecutado nuestro código, se pasa el control a la rutina real OSWRCH por medio de la instrucción JMP con el anterior contenido de OSWRCHV. Después de ejecutar el programa, todas las mayúsculas del programa listadas en la pantalla o digitadas en el teclado se convertirán en minúsculas por obra de nuestra rutina. Al pulsar BREAK, se restablece a OSWRCHV en sus valores normales.

Programa de empleo de memoria

Este listado es otro programa de utilidad, que inspecciona el empleo de la memoria mientras usted está programando.

La expresión:

```
HIMEM-(?2+256*?3)
```

proporciona la cantidad de memoria que queda después de ocupar su espacio el programa y las variables (las posiciones 2 y 3 contienen la dirección del límite superior del cuadro de variables en BASIC). Así la expresión indica cuánta memoria le queda disponible. Pero imprimir (PRINT) una y otra vez este valor sería algo tedioso. La rutina usa el evento Un carácter entra en el buffer de entrada para evaluar la expresión cada vez que se pulsa una tecla, y emite un pitido cuando la memoria restante para el BASIC desciende por debajo de un cierto nivel.

El programa se dispara por medio de un evento de pulsación de tecla y continuará operando entre bastidores mientras usted entra su programa.

La primera tarea del programa es determinar el nivel al cual el usuario desea ser advertido de la memoria que le queda. Esto lo hace PROCselect-memory, que establece el nivel en la forma *hi-lo*. El programa se posiciona en la dirección &0A00, y (como siempre ocurre con programas que emplean eventos) primero guarda el contenido de todos los registros llevándolos a la pila. Después de llamar la subrutina que se encarga de ello, los registros se restauran y se ejecuta RTS para devolver el control del programa allí donde estaba cuando sucedió el evento.

La principal subrutina llama a otras dos subrutinas para calcular la cantidad de memoria que queda. La rutina inc toma el valor almacenado en las posiciones 2 y 3 y le suma el byte *lo* y el byte *hi* del nivel, almacenando el resultado en las direcciones &70 y &71. La rutina sub realiza entonces un cálculo de 16 bits, restando el contenido de &70 y

Uso de la memoria

```
100 REM **** Empleo de evento por pulsacion de tecla ****
110 REM **** para alertar sobre el uso de la memoria ****
120 MODE 1
130 PROCselect_memory
140 PROCassemble
150 END
160 .
170 DEFPROCselect_memory
180 INPUT "Pitido despues de cuantos K restantes",n : n=n*1024
190 hibyte=n DIV 256
200 lobyte=n MOD 256
210 ENDPROC
220
230 DEFPROCassemble
240 oswrch=&FFEE
250
260
270 FOR pass=0 TO 3 STEP 3
280 P%=&0A00
290
300 [OPT pass
310 .start
320 PHP
330 PHA
340 TXA:PHA
350 TYA:PHA
360
370 JSR memory_check
380
390
400 PLA:TAY
410 PLA:TAX
420 PLA
430 PLP
440 RTS
450
460 .memory_check
470 JSR inc
480 JSR sub
490 BPL room
500 LDA #7
510 JSR oswrch
520 .room RTS
530
540 .sub SEC
550 LDA &06:SBC &70:STA &70
560 LDA &07:SBC &71:STA &71
570 RTS
580 .inc CLC
590 LDA &02:ADC #lobyte:STA &70
600 LDA &03:ADC #hibyte:STA &71
610 RTS
620
630 ]:NEXT
640 ?&220=start MOD 256
650 ?&221=start DIV 256
660 *FX14,2
670 ENDPROC
```

&71 de la HIMEM. El valor de HIMEM se almacena en la página cero en las posiciones &06 y &07. Obsérvese aquí cómo se han empleado posiciones de memoria absolutas, principalmente porque estas posiciones han recibido este empleo en todas las versiones del BASIC del BBC.

Una vez efectuada la resta, la línea 490 comprueba el signo del resultado. Si es positivo, indica que hay todavía más bytes de memoria que el nivel previamente determinado. Si es negativo, ya no queda memoria, por lo que se ha de emitir un pitido. La llamada OSWRCH con A=7 es la que se encarga de emitirlo.

Los dos programas hasta aquí analizados son sólo unos sencillos ejemplos de posibles y útiles aplicaciones. Una ventaja del uso de las llamadas del sistema operativo directamente es que podemos producir un código dirigido con eventos, facilidad imposible en BASIC.

Los dos programas que transcribimos a continuación emplean eventos.

Música de fondo

```
 10 REM **** EVENTOS SONOROS ****
 50
 60 PROCassemble
 80 CALL initialise
 90 END
100
110 DEFPROCassemble
120 note_count=&70
130 oswrch=&FFEE
140 osbyte=&FFF4
150 osword=&FFF1
160
170 FOR pass=0 TO 2 STEP 2
180 P%=&0C00
190
200 [OPT pass
210 .initialise
220 LDA #event MOD 256:STA &220
230 LDA #event DIV 256:STA &221
240 LDA #14:LDX #5:JSR osbyte
250 LDA #0:STA note_count
260 JSR clock
270 RTS
280
290 .event
300 PHP
310 PHA
320 TXA:PHA
330 TYA:PHA
340
350 JSR play_note
360 JSR clock
370
380
390 PLA:TAY
400 PLA:TAX
410 PLA
420 PLP
430 RTS
440
450 .play_note
460 LDY note_count
470 .sloop LDA notetable,Y
480 STA soundtable+4
490 JSR sound
500 INC note_count
510 LDA note_count:CPM#5:BNE out
520 LDA #0:STA note_count
530 .out RTS
540
550
560 .clock
570 LDX #time MOD 256
580 LDY #time DIV 256
590 LDA #4
600 JSR osword
610 RTS
620
630 .sound PHA
640 TXA:PHA
650 TYA:PHA
660 LDX #soundtable MOD 256
670 LDY #soundtable DIV 256
680 LDA #7
690 JSR osword
700 PLA:TAY
710 PLA:TAX
720 PLA
730 RTS
740
750 .notetable
760 EQUD &00000000
770 EQUD &00000000
780 .soundtable
790 EQUD &00000000
800 EQUD &00000000
810 .time EQUD &FFFFFCF
820 EQUB &FF
830 ]:NEXT
840 FOR I%=0 TO 4:READ data:?
    (notetable+I%)=data:NEXT
850 FOR I%=0 TO 7:READ data:?
    (soundtable+I%)=data:NEXT
860 ENDPROC
870 DATA 69,73,81,89,97
880 DATA 1,0,&FA,&FF,0,0,10,0
```

Este programa emplea el evento El reloj de intervalos se agotó para hacer sonar una frase de notas musicales mientras el ordenador continúa realizando otras tareas. Dado que la rutina del evento opera como fondo, se puede guardar el programa, editarlo, listarlo o ejecutar otro programa diferente, mientras suenan a cada momento las notas musicales. Se puede ampliar el programa para que suene una melodía mientras se ejecuta, por ejemplo, un programa de juego. La melodía se hará oír incluso durante operaciones con disco.

El programa se ensambló para que se inicie en la dirección &0C00. La primera sección es una rutina de inicialización que realiza las siguientes tareas. El vector de eventos EVENTV, en las direcciones &220 y &221, se cambia para que apunte a nuestra rutina. Se llama entonces a OSBYTE 14 con X=5 para activar el evento El reloj de intervalos se agotó. Un *contador de notas* se inicializa con valor cero en &70 y finalmente el reloj del evento es puesto en marcha con una llamada a la subrutina del reloj.

La primera tarea de la rutina de tratamiento de eventos es guardar en la pila los registros de la CPU. Seguidamente se llama una subrutina que hace sonar una sola nota y se restablece el reloj. Finalmente, se restauran los registros. Nótese que el fragmento del programa que guarda y restaura los registros es común a todas nuestras rutinas de tratamiento de eventos, y basta con cambiar el código entre estos dos bloques de instrucciones.

Examinemos las diferentes partes de la subrutina que hace sonar las notas. Cada vez que el reloj del evento pasa por el cero, suena una única nota, seleccionada de un cuadro de notas. El cuadro de notas, que llamamos notetable, consiste sencillamente en una serie de bytes que contienen la información de la altura de las cinco notas que deseamos tocar. Se emplea un contador para determinar cuál de las cinco notas ha de sonar, reteniendo el valor en &70. El valor de la altura se recoge y almacena en el lugar apropiado dentro de soundtable, que no es más que un bloque de parámetros para la rutina OSWORD que utilizaremos más tarde para producir el sonido. El contador se incrementa entonces, apuntando así a la siguiente posición dentro del cuadro de notas para la siguiente vez que entre en la rutina. Dado que se dispone de tan sólo cinco notas, tan pronto como la posición &70 retiene el valor cinco, el citado cuadro se restablece a cero.

La subrutina clock (reloj) utiliza OSWORD con A=4 para poner en acción el reloj del evento. El área de memoria denominada time retiene el valor que hay que almacenar en los registros del reloj, con lo que se especifica el tiempo que media entre dos notas emitidas. La subrutina sound hace algo semejante a la instrucción SOUND, por medio de OSWORD con A=7; soundtable es el bloque de parámetros para esta llamada OSWORD. Los datos para notetable y soundtable son establecidos por las sentencias en BASIC de las líneas 840 y 850.

Pulsando BREAK se detendrá este programa, restableciendo el contenido de EVENTV. Sin embargo, se puede hacer que continúe llamando a initialise.

El último programa que presentamos ilustra el modo de usar eventos para mover un sencillo dibujo de gráficos, concretamente un rectángulo, a través de la pantalla, una posición por cada evento. La frecuencia de movimiento depende, por ello, del valor asignado al reloj de eventos. Este valor se establece en la línea 1260. La definición del dibujo móvil, en forma de números PLOT, se almacena en shapetable y se establece por una sentencia DATA, mientras la línea 1220 coloca (POKE) los valores apropiados. Las líneas de la 1230 a la 1250 establecen EVENTV para apuntar a nuestro programa y activar el evento adecuado. Examinemos ahora el lenguaje máquina involucrado.

Está almacenado en la memoria en la dirección especificada por code%. Las líneas de la 200 a la 230 realizan la habitual operación de almacenamiento del registro, y las líneas de la 250 a la 280 hacen el trabajo por medio de una serie de llamadas a subrutinas. Finalmente, restauramos los registros y ejecutamos una RTS. Las líneas de la 370 a la 440 guardan el actual estado del cursor de gráficos y el actual GCOL usado, de modo que pueda ser almacenado después de tratado el evento.

De la 460 a la 520 se ocupan del movimiento del

Movimiento de gráficos

```
10 REM **** GRAFICOS CON USO RELOJ EVENTOS ****
30 MODE 1
40 PROCassemble
50 CALL clock
60 END
70 DEFPROCassemble
80 xpos=&70:ypos=&72
90 ?xpos=00:?(xpos+1)=0
100 ?ypos=100:?(ypos+1)=0
110 oswrch=&FFEE
120 osbyte=&FFF4
130 osword=&FFF1
140 DIM code% 600, time% 10
150
160 FOR pass=0 TO 2 STEP 2
170 P%=code%
180
190 [OPT pass
200 PHP
210 PHA
220 TXA:PHA
230 TYA:PHA
240
250 JSR preserve
260 JSR draw
270 JSR restore
280 JSR clock
290
300
310 PLA:TAY
320 PLA:TAX
330 PLA
340 PLP
350 RTS
360
370 .preserve
380 LDX #pgpos MOD 256
390 LDY #pgpos DIV 256
400 LDA #&0D
410 JSR osword
420 LDA &35B:STA ccol
430 LDA &35C:STA ccol+1
440 RTS
450
460 .draw
470 JSR gcol
480 JSR move
490 JSR shape
500 JSR incx
510 JSR incy
520 RTS
530
540 .restore
550 LDA ccol:STA &35B
560 LDA ccol+1:STA &35C
570 LDA #25:JSR oswrch
580 LDA #4:JSR oswrch
590 LDA cgpos:JSR oswrch
600 LDA cgpos+1:JSR oswrch
610 LDA cgpos+2:JSR oswrch
620 LDA cgpos+3:JSR oswrch
630 RTS
640
650 .move
660 LDA #25:JSR oswrch
670 LDA #4:JSR oswrch
680 LDA xpos:JSR oswrch
690 LDA xpos+1:JSR oswrch
700 LDA xpos+2:JSR oswrch
710 LDA xpos+3:JSR oswrch
720 RTS
730
740 .shape
750 LDX#24:LDY #0
760 .sloop LDA shapetable,Y
770 JSR oswrch
780 INY:DEX
790 BNE sloop
800 RTS
810
820 .incx CLC
830 LDA #2:ADC &70:STA &70
840 LDA #0:ADC &71:STA &71
850 RTS
860 .incy CLC
870 LDA #2:ADC &72:STA &72
880 LDA #0:ADC &73:STA &73
890 LDA &73:CMP #3:BCS overflow
900 RTS
910 .overflow
920 LDA #0:STA &72:STA &73
930 LDA #0:STA &70:STA &71
940 RTS
950
960 .clock
970 LDX #time% MOD 256
980 LDY #time% DIV 256
990 LDA #4
1000 JSR osword
1010 RTS
1020
1030
1040 .gcol
1050 LDA #18:JSR oswrch
1060 LDA #3:JSR oswrch
1070 LDA #1:JSR oswrch
1080 RTS
1090
1100
1110 .ccol EQUD &0000
1120 .pgpos EQUD &00000000
1130 .cgpos EQUD &00000000
1140 .shapetable
1150 EQUD &00000000
1160 EQUD &00000000
1170 EQUD &00000000
1180 EQUD &00000000
1190 EQUD &00000000
1200 EQUD &00000000
1210 ]:NEXT
1220 FOR I%=0 TO 23:READ data:?(shapetable+I%)=data:NEXT
1230 ?&220=code% MOD 256
1240 ?&221=code% DIV 256
1250 *FX14,5
1260 !time%=&FFFFFFEF:time%?4=&FF
1270 ENDPROC
1280 DATA 25,1,60,0,0,0,25,1,0,0,60,0,25,1,&C4,255,0,0,
     25,1,0,0,&C4,255
```

dibujo. De la 540 a la 630 restablecen los colores de los gráficos y la posición del cursor de gráficos al estado que tenían cuando entramos en la rutina.

De la 650 a la 720 emplean la rutina OSWRCH para ejecutar la instrucción equivalente a MOVE del BASIC según las posiciones *x* e *y* deseadas. Estas posiciones se almacenan en xpos y xpos+1 para la abscisa *x* e ypos e ypos+1 para la ordenada *y*. Las líneas de la 740 a la 800 dibujan la figura, de nuevo a través de OSWRCH. El registro Y se emplea como registro índice para acceder a la tabla de bytes que representa el dibujo (shapetable). Los bytes se envían como una corriente a través de OSWRCH.

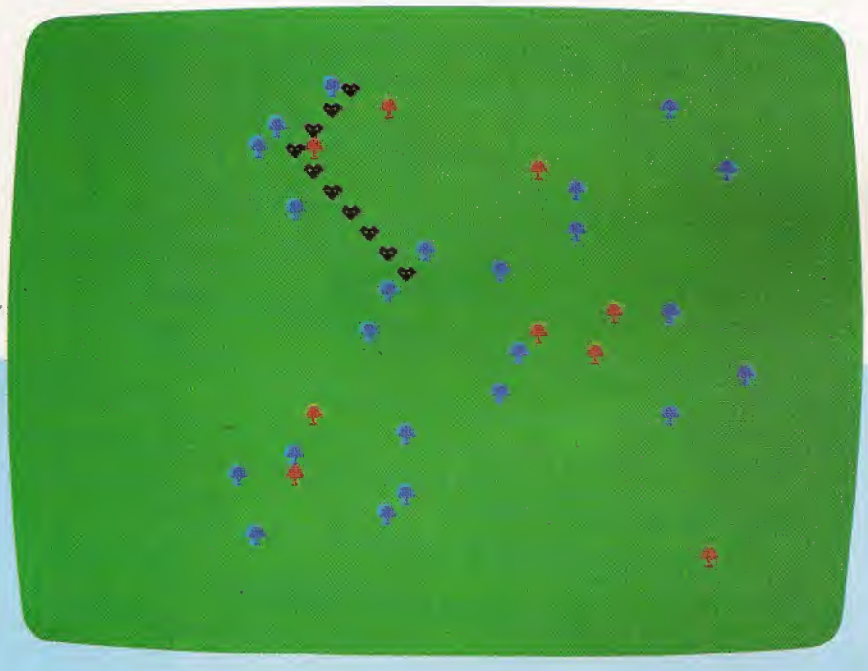
Dado que el dibujo se mueve por la pantalla, es claro que deben existir rutinas para actualizar las posiciones *x* e *y* cada vez que sucede el evento. De esto se encargan las rutinas incx e incy en las líneas de la 820 a la 900. Una vez que la caja alcanza la parte superior de la pantalla, la rutina overflow restablece las coordenadas *x* e *y* a cero. La rutina clock emplea OSWORD con A=4 para accionar el reloj de eventos, y la rutina gcol hace algo equivalente a GCOL3,1 del BASIC.

Hay varios problemas asociados a este programa. Ejecute el código máquina, y siempre y cuando la pantalla no se desplace (*scroll*), el dibujo se moverá bastante bien por la pantalla. Se pueden realizar otras operaciones mientras esto sucede, y todo es correcto. Pero si se acelera el movimiento incrementando el valor del área de memoria time, y aunque el dibujo se mueve cuando se lista parte del programa, puede que aparezcan caracteres extraños en la pantalla. La razón de todo esto es sencilla: el conflicto entre nuestra rutina que es entrada cuando sucede el evento, y el comportamiento habitual del OS cuando se hace el listado. Ambos requieren el empleo de OSWRCH y lo que sucede es que cuando la rutina de listado del OS es interrumpida por nuestro evento, OSWRCH se deja en un estado que no espera la rutina de listado.

Si se interrumpe OSWRCH y es llamada por la rutina de interrupción, entonces no puede obtenerse una salida de la llamada OSWRCH de la rutina de interrupción. Se dice que OSWRCH es *no-reentrante*. Por esto Acorn sugiere que no es recomendable el uso de rutinas OS en las rutinas de tratamiento de eventos y de interrupciones. Pero usted *puede* usarlas, siempre y cuando lo haga con cuidado. Estos problemas pueden ser superados mediante la renuncia al uso de cualquier rutina OS cuando se trate de una rutina de tratamiento de eventos o interrupción. Pero existen alternativas tales como entrar datos directamente en el área RAM de la pantalla para dibujar gráficos, y ésta es una de las pocas veces en que las llamadas de una rutina que evita la ROM es una buena idea.

Serpiente

Esta versión del conocido juego escrita en BASIC para el Thomson MO5 debe grabarse tal cual en cassette, utilizando la instrucción SAVE"<SERPIENTE>"



En este juego, usted es una serpiente que se desliza ondulando por la pantalla. El cambio de dirección se consigue pulsando cualquier tecla. Para poder desplazarse es preciso que se alimente. Felizmente se halla rodeado por gran cantidad de setas. Pero ¡cuidado! Si bien las setas azules son excelentes, ha de evitar las setas rojas, puesto que son venenosas. Cada seta azul le proporciona las calorías necesarias para avanzar diez líneas. ¡Procure no morirse de hambre pero sin acabar envenenado!

```
 10 REM ******************
 20 REM *     SERPIENTE    *
 30 REM ******************
 40 CLEAR ,,2
 50 GOSUB 540
 60 D$=INKEY$
 70 IF D$<>"" THEN D=-D
 80 X=X+D
 90 IF X<0 THEN X=1
100 IF X>39 THEN X=38
110 IF POINT (X*8+4,Y*8+4)=1 THEN 260
120 IF POINT (X*8+4,Y*8+4)=4 THEN S=S+10:
    H=H+10:BEEP
130 LOCATE X,Y
140 COLOR 0
150 PRINT S$;
160 LOCATE 0,24,0
170 PRINT
180 LOCATE INT(RND*38)+1,24
190 COLOR 4
200 PRINT C$;
210 IF RND>0.5 THEN LOCATE INT (RND*38)+1
    ,24:COLOR 1:PRINT C$;
220 S=S-1
230 IF S=0 THEN 260
240 H=H+1
250 GOTO 60
260 BEEP
270 LOCATE 0,24
280 PRINT
290 LOCATE X,Y-1
300 COLOR 5
310 PRINT S$;
320 FOR I=1 TO 5
330 BEEP
340 FOR J=1 TO 50
350 NEXT J
360 NEXT I
370 IF INKEY$<>"" THEN 370
380 IF H>R THEN R=H
390 LOCATE 4,20
400 COLOR 0
410 PRINT "PUNTOS :";H;
420 LOCATE 23,20
430 PRINT "RECORD :";R;
440 LOCATE 15,23
450 PRINT "OTRA ?"
460 D$=INKEY$
470 IF D$="" THEN 460
480 IF D$<>"N" THEN 520
490 SCREEN 4,6,6
500 CLS
510 END
520 GOSUB 610
530 GOTO 60
540 CLS
550 SCREEN 2,2,2
560 DEFINT A-Z
570 DEFGR$(0)=0,0,102,255,219,126,60,24
580 DEFGR$(1)=60,126,126,255,255,24,24,
    60
590 S$=GR$(0)
600 C$=GR$(1)
610 CLS
620 S=100
630 H=0
640 D=1
650 X=19
660 Y=10
670 RETURN
```

Lenguajes modernos

¿Cuáles son las causas de la controversia existente respecto a enseñar programación de ordenadores a alumnos de escuela primaria?

El papel tradicional del niño en la educación es el de receptor del aprendizaje. El conferirle un papel activo suscita una controversia inevitable, por lo que no debe resultar sorprendente el que educar al niño en el campo de la programación sea una tarea de elevado riesgo tanto para los maestros como para los alumnos.

Los estudios de informática se incorporaron a los planes de estudio a finales de los años setenta y han ido adquiriendo una creciente aceptación entre los alumnos y los padres, indistintamente. Encuestas realizadas en Gran Bretaña entre los padres a comienzos de los ochenta revelaron que éstos consideraban los estudios de informática como la tercera asignatura escolar en escala de importancia (tras matemáticas e inglés); pero este entusiasmo no siempre es compartido por los miembros del cuerpo de profesores, ni tampoco por las personas pertenecientes a la industria del ordenador, a la caza de nuevos adeptos.

Algunos maestros, por ejemplo, piensan que los ordenadores deberían integrarse en todas las áreas de los planes de estudio en vez de otorgárseles un status separado, dominio exclusivo de un Departamento de Estudios Informáticos. Pero en algunas ocasiones, esto ha significado que la ciencia informática, como disciplina, se haya visto afectada por algunos problemas, surgidos, especialmente, a consecuencia de los escasos recursos económicos de que disponen la mayoría de las escuelas. A pesar de ello, muchos especialistas de la educación continúan pensando de forma muy apasionada en los beneficios que reporta la programación *per se*.

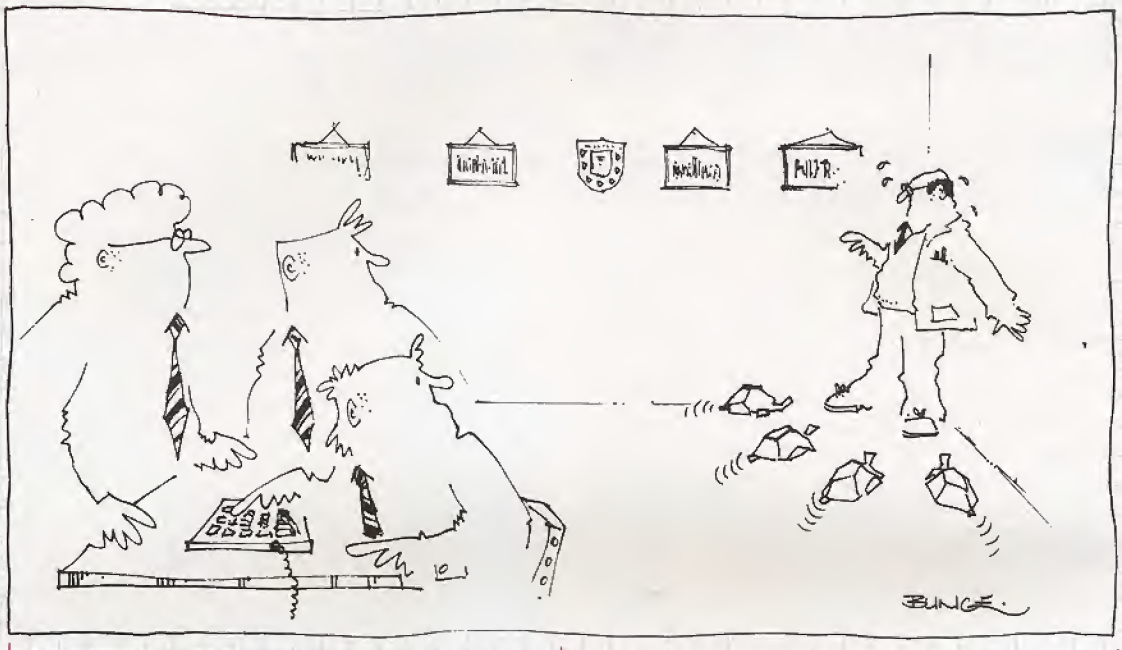
El hecho de que un niño no llegue jamás a convertirse en un artista o un escritor profesional, no justifica la omisión de estos estudios. Como ya hemos visto a lo largo de esta serie, a los niños pequeños se los ayuda a aprender conceptos y habilidades normalmente consideradas fuera de su comprensión como un derivado de la programación, y estos "beneficios colaterales" deben tener por sí mismos alguna justificación, al menos para enseñar programación en las escuelas.

Hasta hace poco tiempo, la enseñanza de programación en las escuelas se ha visto influida en gran medida por el hecho de que la mayoría de las máquinas se suministran con BASIC. Debido a su contenido abstracto, el BASIC está fuera del alcance de la mayoría de los niños menores de 12 años, de modo que en algunos casos el estudio de informática se ha limitado a las escuelas de nivel secundario (a partir de sexto curso).

Opciones de lenguajes

La implementación de diferentes lenguajes en los micros poco a poco está ayudando a ensanchar la base de los estudios de programación en las escuelas. Algunos lenguajes, como el PROLOG, se pueden estudiar provechosamente sobre la base de su propia estructura. Otros, como el COMAL y el LOGO, son valiosos porque ofrecen una introducción más disciplinada a la programación que el BASIC (que, aunque es fácil de aprender, también puede propiciar una utilización abusiva y llevar a técnicas de programación descuidadas).

Alan Adler



Además, a pesar de su utilidad en otros campos, la industria informática generalmente se muestra reticente respecto al empleo del BASIC en la educación, debido a su enfoque no estructurado y a la forma en que puede propiciar la adquisición de malos hábitos de programación en el principiante. Lamentablemente, muchos maestros del área de estudios informáticos se han introducido por propia iniciativa en el tema mediante cortos cursillos de BASIC, propagando de ese modo su influencia a pesar de la disponibilidad de otros lenguajes, como el LOGO, COMAL y PROLOG.

Si, como se ha sugerido ampliamente, el empleo del BASIC favorece malas técnicas de programación y un pensamiento desordenado o abstracto, ¿cuáles son las alternativas?

La introducción de versiones más estructuradas del BASIC (el BASIC BBC, p. ej.) ha ayudado a atenuar algunos recelos industriales, pero el verdadero interés por la programación educativa reside en los tres lenguajes que hemos mencionado anteriormente. A lo largo de esta serie ya hemos visto con cierto detalle el LOGO. Se está promoviendo de forma activa en Gran Bretaña y Estados Unidos a través del British LOGO Users Group y la Young Persons LOGO Association, respectivamente. Todos los fabricantes de micros populares han sacado a la venta versiones del lenguaje, rescatándolo de la oscuridad de los proyectos de investigación de inteligencia artificial, en los que en algún momento pareció destinado a quedar confinado.

En el State Teacher's College, de Tonder (Dinamarca), Borge Christensen desarrolló el COMAL (COMmon Algorithmic Language). El COMAL, según su creador, es una forma estructurada del BASIC. Christensen y sus colegas se encontraban con que los programas en BASIC de sus alumnos estaban tan mal escritos que eran casi incomprensibles. "Tras procesar cantidades ingentes de programas en BASIC —comentó—, comprendí que algo olía a podrido en Dinamarca." Los argumentos de Christensen en favor del COMAL han ejercido influencia en el gobierno de Dinamarca, al punto de que se ha adoptado el COMAL como primer lenguaje de programación en las escuelas danesas. También se lo prefiere, en detrimento del BASIC, en la República de Irlanda, y en Gran Bretaña ha despertado cierto interés. En una escuela secundaria del sur de Londres se inició un ambicioso proyecto en el cual los alumnos debían escribir un programa de localización de satélites en COMAL.

Un pequeño grupo de especialistas en educación de Inglaterra ha estado utilizando el PROLOG (PROgramming in LOGic). Empleando el PROLOG, un problema expresado en términos lógicos se puede traducir fácilmente a un programa para ordenador. En Park House School (Londres) se planificó un proyecto para desarrollar material para usar en la enseñanza de lógica como una disciplina de programación para alumnos de entre 10 y 13 años. Algunos de éstos aprendieron a programar fluidamente y otros, aunque participaron en los trabajos de la clase, escribieron poco por sí mismos. Con posterioridad, a medida que el conocimiento de la informática se generalizó en toda la escuela, en las clases posteriores se consiguió un mayor éxito. El proyecto le ha dado más importancia a la enseñanza de la lógica que a la de programación.

¿Cuáles son las diferencias existentes entre estos lenguajes que podrían determinar lo que aprenda el niño a partir de la programación?

El LOGO ha sido objeto de profundas investigaciones, en un intento por dar una respuesta a los críticos que afirman que existen pocas pruebas de los beneficios que obtienen los niños a través de la programación en este lenguaje. (Lamentablemente, la relativa minoría de escuelas que utilizan COMAL y PROLOG impide que estos len-

guajes se puedan estudiar adecuadamente.) En Estados Unidos, la National Science Foundation financió el proyecto de LOGO Brookline para estudiar el empleo del LOGO por parte de los niños. La primera fase del proyecto implicó poner a los alumnos de sexto curso (11 años de edad) en un laboratorio de ordenadores y observar sus características personales y sus actividades. En la segunda fase los ordenadores se introdujeron en la escuela y se trabajó parcialmente desarrollando material para los planes de estudio. Un resultado interesante de la primera fase fue que todos los niños realizaron con éxito su trabajo en LOGO, independientemente de la capacidad demostrada en otras asignaturas. Este mismo resultado se ha producido en otros proyectos de LOGO.

¿Qué otros intentos se han realizado para estudiar los efectos en niños que han aprendido a programar?

La Academia de Ciencias de Nueva York y tres escuelas de esta misma ciudad prepararon un proyecto para dar entrenamiento y apoyo a la enseñanza del LOGO en las escuelas. En todas las aulas se instalaron ordenadores, miembros del proyecto visitaron semanalmente las escuelas, y maestros voluntarios participaron en seminarios mensuales. El personal calculó que el éxito del proyecto alcanzó a casi el 95 % de las aulas. El director del proyecto señaló numerosas similitudes con el proyecto Brookline. Una consecuencia fundamental de disponer del LOGO en la clase fue la positiva interacción entre los estudiantes. Los niños que no habían obtenido resultados satisfactorios en otras asignaturas se desenvolvieron muy bien con el LOGO. "El beneficio educativo fue evidente —manifestó—. Los niños se comprometieron en procesos intelectuales y sociales válidos. Hemos observado cambios notables en las relaciones de los niños con la escuela y con el aprendizaje." Un producto del proyecto fue la recomendación de contar en la zona con una "escuela imán" que actuara como eje para las investigaciones centradas en el LOGO.

En el Center for Children and Technology de Nueva York se llevó a cabo un proyecto de 30 meses para evaluar la transferencia de estrategias de resolución de problemas a otras áreas de los planes de estudio, y para juzgar el efecto del trabajo orientado hacia el LOGO sobre la interacción social.

Es interesante observar la forma en que los ordenadores han estimulado la cooperación e interacción entre los niños. "Durante la actividad con el ordenador hubo más interacción en la tarea que durante cualquier otra actividad de la clase no dirigida por el maestro", observó el jefe del programa.

¿Qué problemas, si los hubiera, han obstaculizado la eficacia de estos estudios?

Son muchísimos los problemas que surgen al investigar la efectividad de la programación en las escuelas. Los pedagogos se muestran reacios a asignar mucho tiempo a la programación cuando los beneficios de su inclusión en los planes de estudio no parecen evidentes. Los fondos de financiación y, por consiguiente, el hardware son escasos. También es difícil llevar a cabo un estudio controlado, puesto que la creación de un grupo de control en la misma escuela implica privar a otros niños de horas frente al ordenador, mientras que tener un grupo de control en otra escuela dificulta la comparación, debido a los diferentes programas de estudio y métodos de enseñanza. Muchos maestros están tan poco familiarizados con los ordenadores como los propios niños, de modo que con frecuencia éstos no disponen de ninguna fuente de conocimientos a la cual recurrir.

Aparte de desarrollar enfoques más lógicos, ¿qué otras aplicaciones prácticas puede esperar hallar el niño después de aprender a programar?

Una vez que el niño se ha introducido en el arte de la programación, la cantidad de aplicaciones potenciales es inmensa. Un grupo de niños ha utilizado tortugas para coreografía; habiendo programado éstas para desplazarse en formación, transformaron luego los movimientos de los dispositivos en danza. Las características del LOGO para la programación de palabras y listas se han empleado para generar historias en las clases de lengua y literatura. Se les entrega a los niños una serie de bifurcaciones que ilustran las diversas opciones que determinan la dirección del cuento. Además, el LOGO posee amplias facilidades de sonido. De la misma forma en que se utiliza CUADRADO para definir PATRON, NOTA puede definir ACORDE, ACORDE puede definir MELODIA, y así sucesivamente, permitiendo, por consiguiente, que los niños integren su uso del ordenador con sus estudios de música.

Si son pocas las escuelas que enseñan programación, y en lenguajes diferentes, ¿cómo podrían extenderse los posibles beneficios?

En Gran Bretaña, al menos, el continuo establecimiento del LOGO como el lenguaje de programación para los niños pequeños ha supuesto la aparición de nuevas e interesantes áreas de actividad del plan de estudios. Pero al nivel secundario, el BASIC continúa disfrutando de su dominio y es necesario poner en práctica algunas medidas para asegurar que el continuo apoyo que se presta al LOGO se extienda a todo el sistema educativo. De lo contrario, la confusión que rodea al estudio de informática en las escuelas probablemente irá en aumento, cuando los programadores cualificados de LOGO vayan subiendo de nivel en el sistema y se encuentren con que han de abordar el BASIC. Tal como sucede con los ordenadores, su empleo en las escuelas también se debe estandarizar para que pueda producir beneficios a largo plazo.

Aprender a programar le ofrece al niño otro medio de expresión y creatividad y le estimula a pensar en una forma precisa y organizada.

Digitando

En nuestra serie sobre bases de datos vamos a examinar ahora las "claves" primarias y secundarias

Una *clave* es una herramienta incorporada en el software del administrador de bases de datos que ayuda al usuario a localizar un registro específico. Consideremos el manual de mantenimiento de un coche. Si usted desea ajustar el carburador, la información relativa a la forma de hacerlo estará oculta en algún sitio del manual. Podría estar, por ejemplo, en la página 36, de modo que este número de página se podría considerar como la clave para la información que necesita. Sin embargo, lo más probable es que no sepa que del carburador se habla en la página 36, de modo que lo que deberá hacer es remitirse al índice. Éste le proporcionará el número de página que necesita, permitiéndole acudir a la parte correcta del manual sin tener que ir pasando página por página.

Y lo mismo sucede con un DBM. Cada registro de un archivo de base de datos tiene un número de registro exclusivo (conocido por *clave primaria*), pero para hallar un registro específico usted tiene la alternativa de mirar cada registro en secuencia hasta hallar el buscado o, si ya sabe el número de registro, ir directamente al registro en cuestión. Asimismo, también tiene la alternativa de utilizar uno de los campos como clave (denominado técnicamente *clave secundaria*). Si tuviéramos una base de datos sobre servicio de coches, emplearíamos el campo NOMBRECOMPONENTE como clave.

La mayoría de los DBM permiten designar campos específicos como "campos clave". Cuando un campo (NOMBRECOMPONENTE, en este caso) se ha designado como un campo clave, el DBM mantiene una tabla interna de palabras (series de caracteres) del campo especificado, junto con el número de registro apropiado (clave primaria). Cuando usted busca el registro sobre el carburador, el DBM busca en la tabla NOMBRECOMPONENTE hasta encontrar la serie de caracteres "carburador", ve cuál es el número de registro adecuado y después extrae ese registro. He aquí cómo podría implementar tal esquema en un DBM simple en BASIC:

```
INPUT'ENTRE CAMPO CLAVE';CCLAVE$
INPUT'ENTRE PALABRA A BUSCAR';PAL$
GOSUB 20000:REM SUBRUTINA DE BUSQUEDA
PRINT REGRESULT
```

Todo cuanto está sucediendo aquí es que se ha seleccionado una de varias matrices, utilizando CCLAVE$ y la subrutina ha efectuado la búsqueda empleando la serie PAL$ como una clave de búsqueda. Si la subrutina de búsqueda es suficientemente potente, será capaz de admitir pequeños errores de digitación y aun así hallar un registro probable. Una rutina sencilla como ésta no necesita depender de que se mantenga una tabla de entradas clave: bastarán procedimientos normales de búsqueda a través de todos los registros.

En el pasado, los DBM que han operado en microordenadores han sido bastante simples, como el que podría escribir usted en BASIC en un tiempo moderadamente corto. No obstante, el advenimiento del Sinclair QL, con su procesador 68000, supuso una importantísima brecha. Psion Ltd, utilizando potentes miniordenadores VAX, desarrolló el *Archive* para el QL y puso al alcance del usuario de un micro personal capacidades avanzadas para administración de bases de datos. Para ver cómo trabaja *Archive*, vamos a crear una sencilla base de datos sobre mantenimiento de coches, con apenas cuatro registros. Cada uno se compondrá de sólo dos campos, NOMBRECOMPONENTE y SERVICIO:

```
Carburador
Quitar cubierta y girar perilla

Depósito de aceite
Levantar tapa y llenar con aceite

Batería
Abrir tapa y llenar con agua destilada

Radiador
Abrir tapa y llenar con agua
```

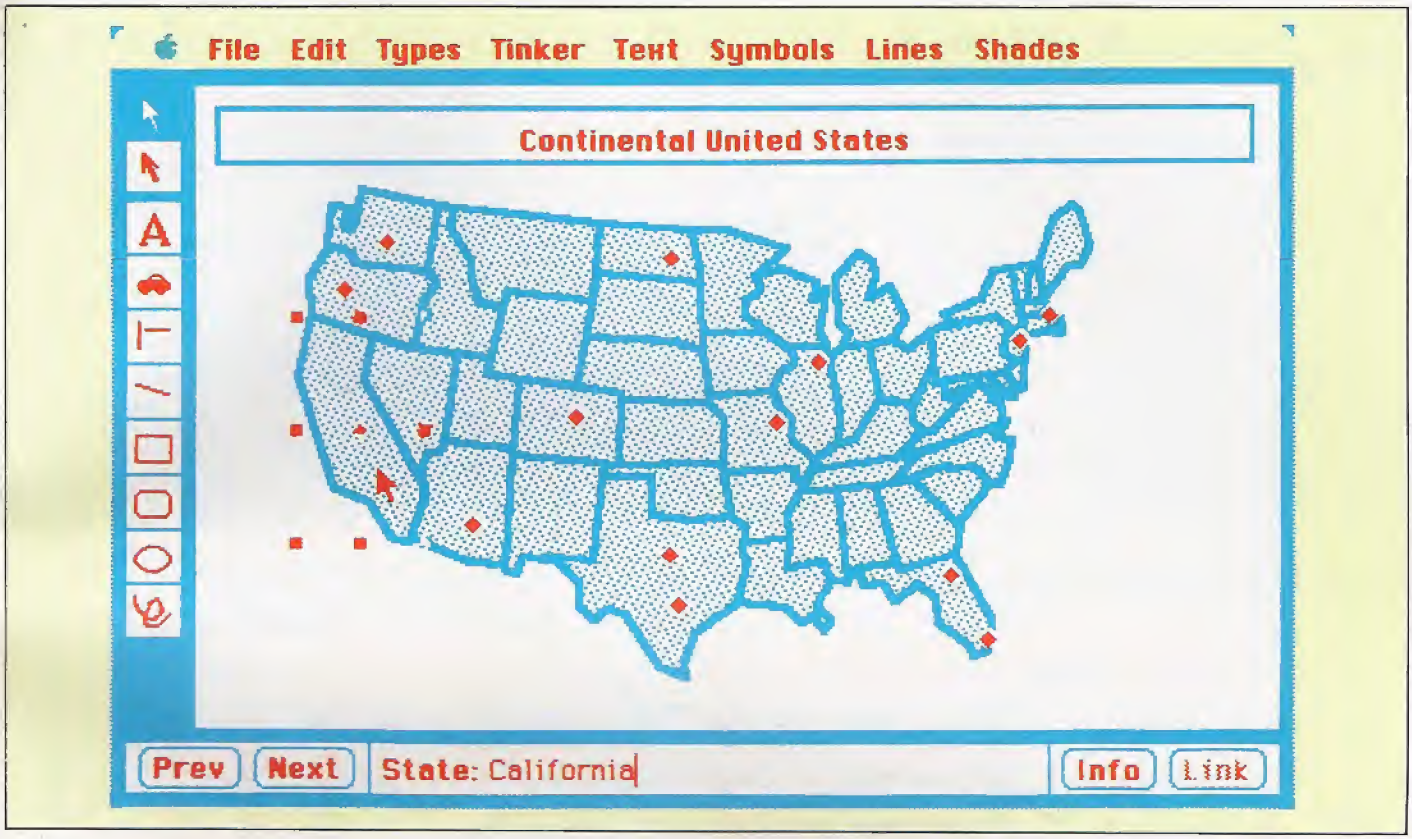
Cuando se ejecuta *Archive* en el QL, al usuario se le presenta una pantalla dividida en tres zonas: una de petición de instrucciones en la parte superior de la pantalla, otra de trabajo en el centro y una tercera de visualización en la parte inferior. Para comenzar un nuevo archivo de base de datos, seleccione la instrucción CREATE. Para crear nuestra base de datos de mantenimiento de automóviles, digite CREATE'COCHE' <CR> ('COCHE' es el nombre que le damos al archivo). Luego entre los nombres de los campos:

```
NOMBRECOMPONENTE$ <CR>
SERVICIO$ <CR>
<CR>
```

El signo $ añadido a los nombres de los campos indica que el campo está compuesto por una serie de caracteres. El <CR> final termina el proceso.

Para añadir registros a la base de datos, se digita la instrucción INSERT<CR>. Ésta visualiza los nombres de campos en la zona de trabajo y permite entrar los datos para cada campo. La pulsación de F5 después de haber entrado correctamente un registro hace que éste se añada al archivo de la base de datos. Una vez digitados todos los registros que desea entrar, puede salir de la modalidad INSERT pulsando la tecla Escape. Se puede entonces salir del archivo digitando la instrucción CLOSE.

Para buscar un archivo *Archive* para un registro, primero debe ser abierto. Esto se puede hacer digitando ya sea la instrucción LOOK, que sólo permite examinar los registros, o bien OPEN, que permite buscar registros y modificarlos. Habiendo abierto un archivo con OPEN o con LOOK, usted puede



entrar instrucciones simples tales como FIRST (para ver el primer registro del archivo), LAST (para ver el último registro), NEXT (para ver el registro siguiente) o BACK (para ver el registro anterior).

La búsqueda de un registro implica añadirle a la instrucción un argumento, como en FIND 'CARBURADOR'. Esta instrucción buscará todos los campos hasta localizar una entrada que concuerde con la parte entre comillas. Asimismo, se pueden utilizar operadores lógicos, como en SEARCH NOMBRECOMPONENTE$='CARBURADOR' AND SERVICIO$='GIRAR'. Esta instrucción buscará en el archivo hasta hallar un registro que contenga la serie CARBURADOR en el campo NOMBRECOMPONENTE$ *y* una serie GIRAR en el campo SERVICIO$ del mismo registro. El operador OR se puede emplear de la misma manera, de modo que se localizará un registro si se puede emparejar la serie especificada para un campo *o* (OR) la serie especificada para el otro campo.

Los registros generalmente se entran en una base de datos de forma caprichosa, pero con frecuencia es necesario acceder a ellos o imprimirlos de acuerdo a cierto orden. Supongamos, por ejemplo, que se le ha encargado la tarea de entrar en una base de datos todos los libros de una biblioteca por autor, título, editorial e ISBN (International Standard Book Number: número de libro según estándar internacional). La forma más sencilla de hacer esto sería ir trabajando a lo largo de los estantes, entrando los datos para cada libro a medida que éstos le van llegando a las manos. Utilizando las capacidades para clasificación que tenga incorporadas su DBM, podría luego clasificar los registros por orden alfabético (por autor, título, etc.), o incluso numéricamente según el ISBN.

Se puede, además, combinar una clasificación primaria con una clasificación secundaria. Supongamos que usted quisiera imprimir el archivo por orden alfabético de autor, con las entradas para cada autor dispuestas por orden alfabético de títulos. Ello permitiría salidas impresas como ésta:

```
CORTAZAR, JULIO
   Octaedro
   Alianza Editorial, S. A.
   84-206-3010-1
CORTAZAR, JULIO
   Queremos tanto a Glenda
   Alfaguara, S. A.
   84-204-2126-X
CORTAZAR, JULIO
   Rayuela
   Bruguera, S. A.
   84-020-6740-9
CORTES, HERNAN
   Cartas de relación de la conquista de México
   Espasa-Calpe, S. A.
   84-239-0547-0
```

Si los nombres de campos del *Archive* fueran AUTOR$, TITULO$, EDITORIAL$ e ISBN, clasificar los registros primero por autor y luego por título sería tan simple como:

```
ORDER AUTOR$;A,TITULO$;A
```

En este ejemplo, la A significa "listar por orden alfabético ascendente".

A estas alturas, ya resulta evidente que los DBM pueden convertir en un trabajo sencillo tareas que consumen tanto tiempo, como son la localización de registros y su reordenamiento de formas específicas. Próximamente analizaremos algunas de las facilidades más avanzadas que tienen incorporadas muchos administradores de bases de datos.

Estados Unidos al dedillo
Filevision se desarrolló para los Apple II y III, pero tuvo que esperar a que el mayor poder de proceso del Macintosh permitiera apreciar sus cualidades. Utiliza la interface para ratón del Macintosh para acceder a la base de datos "señalando y apretando". Al "frente" del programa hay un sofisticado paquete de dibujo que le permite crear una estructura de gráficos para su base de datos. La forma y los símbolos que usted dibuje se conectan a los campos de la base de datos. En la ilustración vemos la pantalla principal para una base de datos que contiene información relativa a Estados Unidos. Si señalamos el ratón para indicar California, un simple clic proporciona una breve cantidad de datos sobre ese estado; un doble clic abre un campo de texto que puede almacenar una cantidad de información muchísimo mayor. Los datos almacenados se pueden enlazar y clasificar de la misma forma que en cualquier base de datos normal. Por ejemplo, escogiendo "Highlight Some" del menú "Tinker", el programa le permitirá especificar un criterio de búsqueda y después visualizar los resultados dibujando con un trazo más grueso los contornos de esos estados en el mapa. Esta información también se puede imprimir seleccionando "Print" después de entrar el criterio especificado

Para asir

Llegados a este punto, nos corresponde diseñar el mecanismo de prensión y las conexiones necesarias para la "musculatura" del robot

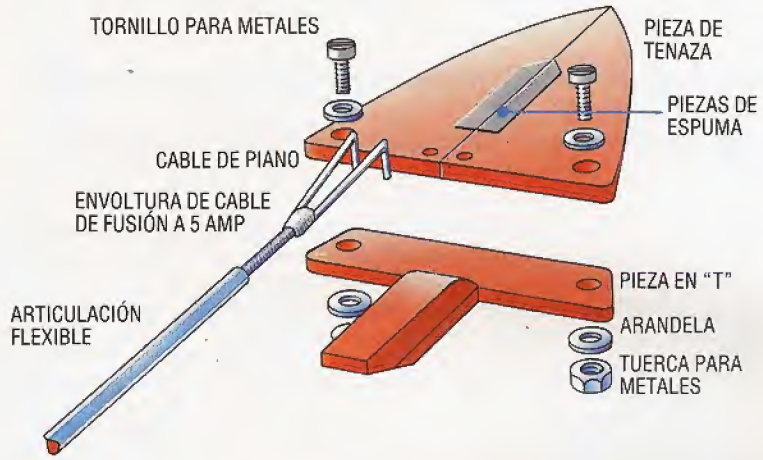
Paso 1: Ensamblaje de la pinza

Taladre agujeros en los centros señalados en la pieza en "T" y las piezas de tenaza para poder colocar los tornillos. Pegue tiras de espuma en la quijada de las piezas de tenaza a modo de relleno. Se deben perforar otros dos agujeros en las posiciones señaladas en los cortes del patrón, de modo que por el agujero quepa estrechamente un trozo de cable de piano rígido. Las piezas de tenaza se pueden ajustar a la pieza en "T" mediante los tornillos y las tuercas para metal. Inserte arandelas arriba y abajo de las piezas de tenaza de modo que cuando se ajusten los tornillos las piezas de tenaza queden libres para pivotar alrededor del tornillo. Tomando un extremo de la articulación flexible de 500 mm, fije dos trozos de 30 mm de cable de piano al cable central envolviendo los extremos del cable de piano y el cable central con alambre de fusión a 5 amp y sellándolo con soldadura. Incline hacia abajo los otros extremos de los cables de piano, formando ángulos rectos, y empújelos en los agujeros preparados en las piezas de tenaza. El cable central y los cables de piano deben formar una "Y". Con las quijadas de la pinza cerradas, ajuste la carcasa de la articulación con algodón y péguela en su sitio.

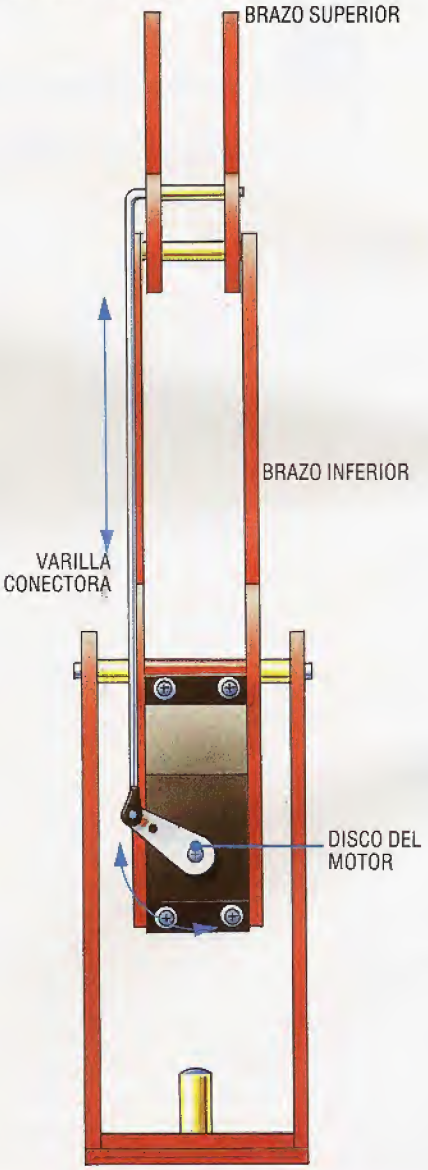
Paso 2: Varilla conectora del brazo sup.

El brazo superior es controlado por el motor ya colocado en el ensamblaje del brazo inferior. El movimiento de rotación del husillo del motor se ha de convertir en un movimiento de tirar-empujar en el brazo superior. Ello se consigue ajustando un corto brazo de plástico (o "bocina") o un disco plástico de 30 mm al husillo. Ambos poseen un agujero por el cual se introducirá la varilla conectora. Aquí asumiremos que se están utilizando discos, pero el sistema funcionará perfectamente si en su lugar se emplean bocinas. Coloque el disco de 30 mm o la bocina en el husillo de modo que, cuando el motor

Paso 1: Ensamblaje de la pinza



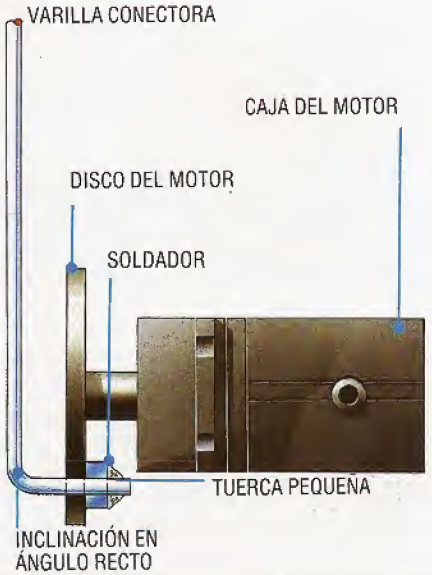
Paso 2: Varilla conectora del brazo superior



Conexión de las varillas a los discos

En el lugar donde adquiera los servomotores podrá también comprar varillas conectoras especiales. Si no las consigue, la mejor forma de conectar una varilla de acero de 2 mm normal al disco del motor consiste en efectuar una inclinación en ángulo recto cerca del extremo de la varilla, pasándola a través del agujero del disco y soldando una pequeña tuerca en la parte posterior del disco. Esto se puede realizar con más facilidad con el disco fuera del husillo del motor.

Donde las varillas se colocan en agujeros de los miembros de madera del brazo, es una buena idea alinear el interior del agujero con una delgada pieza de tubo de cobre de un diámetro interno apenas más grande que el de la varilla conectora. Ello reducirá la abrasión de la madera cuando el brazo se mueva



Kevin Jones

gire completamente en sentido antihorario, el agujero esté justo a la izquierda de la posición de las seis en punto (imagínese que el disco del motor es un cuadrante con las doce en punto situadas verticalmente hacia arriba). Tome un trozo de varilla de acero de 2 mm de diámetro e incline los extremos formando ángulo recto de modo que la varilla quepa a través del agujero del disco del motor o la bocina y los dos agujeros preparados para ella en las piezas del brazo superior, cerca de la juntura del codo. La longitud de la varilla debe ser tal que los brazos superior e inferior formen un ángulo de 90° en el codo cuando el motor gire completamente en sentido antihorario. Compruebe la acción del brazo girando lentamente el disco del motor, a mano, en sentido horario. Pegue o suelde la varilla en su sitio en el disco.

Paso 3: Varilla conectora del brazo inferior

Monte los motores restantes en el cuerpo principal, como indica la ilustración. El motor de la derecha controla el movimiento del brazo inferior. Gire el motor de manera que quede por completo en sentido antihorario (visto desde el lado del disco del motor) y posicione el disco del motor de modo tal que el agujero de la circunferencia del disco quede justo a la derecha de la posición de las seis en punto. Ajuste un trozo de varilla de acero de 2 mm entre el disco del motor y los agujeros del brazo inferior que se hallan cerca de la juntura del hombro, como antes. La longitud de la varilla debe ser tal que el brazo inferior quede aproximadamente horizontal con el motor girado totalmente en contra de las agujas del reloj. Verifique la acción del brazo inferior haciendo girar a mano el disco del motor y pegue o suelde la varilla conectora al disco del motor.

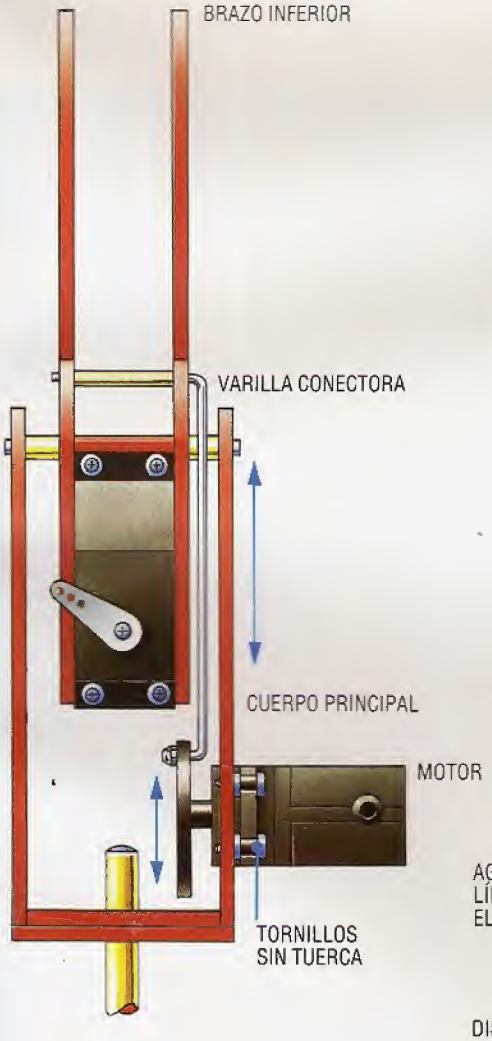
Paso 4: Conexión de la artic. al motor

El motor montado a la izquierda del cuerpo principal se utiliza para abrir y cerrar la pinza, empujando o tirando el cable central de la articulación flexible. Comience por taladrar un agujero a través de la pieza cruzada del cuerpo principal de modo que, como se ve en la ilustración, la articulación flexible se pueda deslizar a través del mismo desde atrás. Pegue la carcasa exterior de la articulación en este agujero. Cuando la cola esté seca, pase el cable central por este agujero, de modo que las quijadas de la pinza queden bien cerradas, y ajuste el extremo del cable al disco del motor. El agujero de la circunferencia del disco debe estar en las "nueve en punto" cuando el motor gire totalmente en sentido antihorario. El extremo del cable se puede fijar al disco del motor de varias maneras. Lo mejor es montar en el disco un pequeño prensacables. De lo contrario, se puede realizar una inclinación de 90° en el extremo del cable, y pegar o soldar en su sitio el extremo que se ha pasado a través del agujero del disco. Independientemente de cómo efectúe esta conexión, lo importante es que el motor pueda empujar y tirar del cable mientras vaya girando.

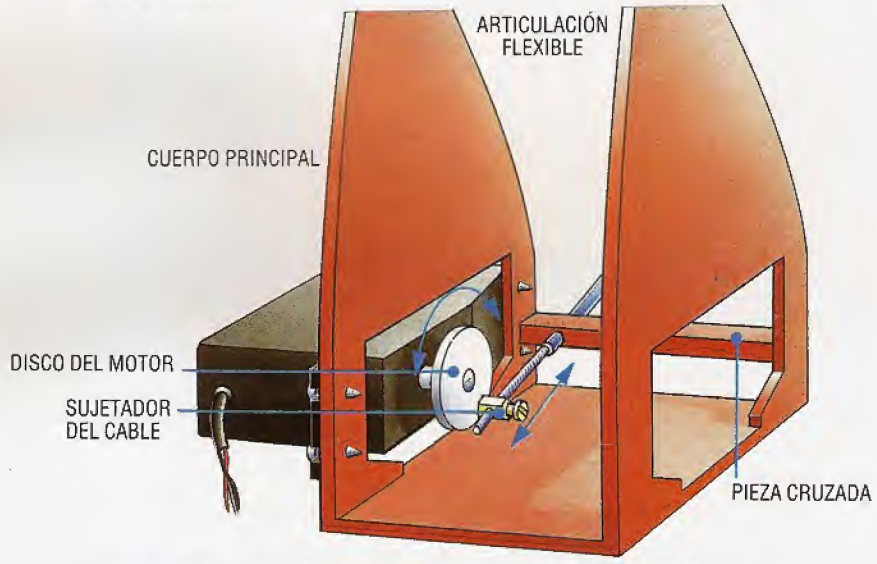
Paso 5: Varilla del cuerpo principal

La última conexión a realizar es la del motor montado en la caja base al ensamblaje del cuerpo principal. Taladre un pequeño agujero en la base del cuerpo principal (necesitará correr el ensamblaje del tarugo de acero). El agujero se debe situar en línea con el pivote, junto al borde derecho (visto desde arriba) de la base del cuerpo principal. Realizado el agujero, vuelva a colocar el ensamblaje del cuerpo principal en el tarugo y gire hasta la posición de las dos en punto. Gire el motor por completo en sentido antihorario y posicione el disco de modo que el agujero de la circunferencia quede justo a la derecha de las seis en punto. Coloque una varilla de conexión como antes y compruebe que el cuerpo principal quede libre para girar en sentido antihorario.

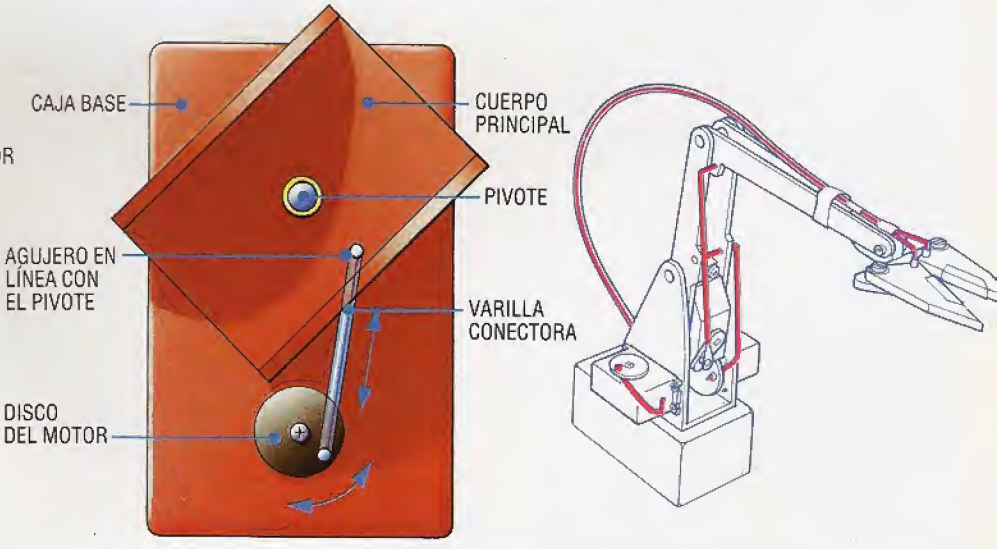
Paso 3: Varilla conectora del brazo inferior



Paso 4: Conexión de la articulación al motor

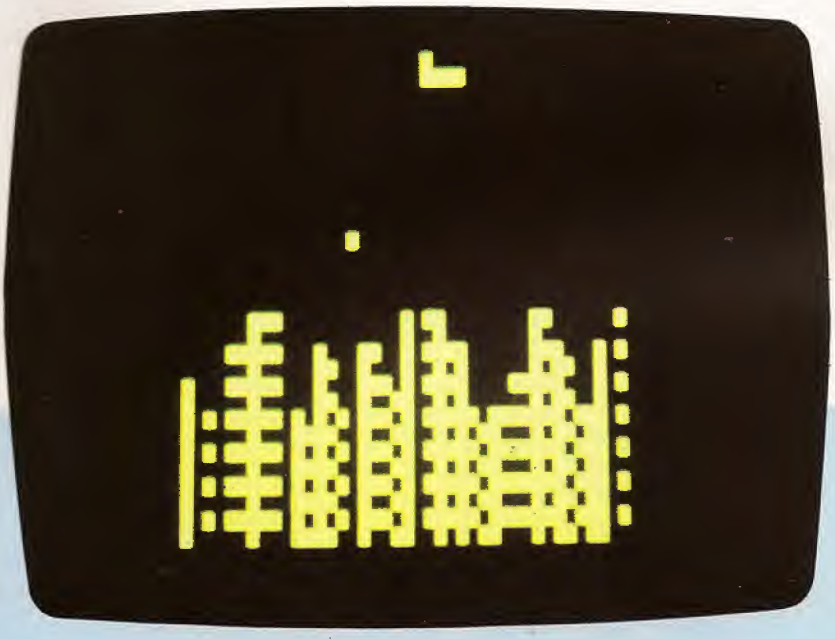


Paso 5: Varilla del cuerpo principal



Bombardeo aéreo

La mayoría de los microordenadores permiten jugar al "bombardeo aéreo" u otros juegos parecidos que se conocen bajo otra denominación. El listado que ofrecemos es para el Dragon



Su misión consiste en destruir la ciudad que está sobrevolando, con objeto de poder aterrizar. A cada pasada, su avión vuela un poco más bajo. No puede lanzar una bomba (pulsando sobre una tecla cualquiera) hasta que la bomba precedente haya alcanzado su objetivo o el suelo. Cuando su avión ha aterrizado (o cuando se ha estrellado contra un edificio), se registra la puntuación, así como el récord del día. Si este juego le parece demasiado difícil, puede variar los límites de la ciudad (6 y 26, línea 100).

```
 10 REM ********************************
 20 REM *          BOMBARDEO AEREO      *
 30 REM ********************************
 34 REM
 35 REM INICIALIZACION
 36 REM
 39 REM A$=AVION
 40 A$=CHR$(128)+CHR$(155)+CHR$(147)
 49 REM B$=BOMBA
 50 B$=CHR$(145)
 58 REM H=POSICION DEL AVION Y
 59 REM PUNTUACION
 60 H=0
 69 REM B=POSICION DE LA BOMBA
 70 B=0
 80 B1=B
 90 CLS 0
 94 REM
 95 REM COLOCACION DE LA CIUDAD
 96 REM
 99 REM 6 y 26: LIMITES DE LA CIUDAD
100 FOR I=6 TO 26
110 C=RND (7)+151
114 REM LINEA 120:CAMBIAR 8
115 REM POR UN VALOR SUPERIOR
116 REM PARA DISMINUIR LA ALTURA
117 REM DE LAS CASAS
120 FOR J=15 TO RND(4)+8 STEP-1
130 PRINT@J*32+I,CHR$(C);
140 NEXT J
150 NEXT I
154 REM
155 REM BUCLE PRINCIPAL
156 REM
159 REM COLOCACION DEL AVION
160 PRINT@H,A$;
169 REM AVION ESTRELLADO?
170 IF PEEK(1027+H)<>128 THEN 240
179 REM LANZAMIENTO
180 IF INKEY$<>"" AND B=0 THEN B=H+33
189 REM BOMBA QUE LLEGA AL SUELO
190 IF B<>0 THEN GOSUB 360
199 REM NO SE HA SOLTADO LA BOMBA?
200 IF B=0 THEN GOSUB 400
209 REM AVANCE DEL AVION
210 H=H+1
219 REM AVION ATERRIZADO
220 IF H=507 THEN 240
230 GOTO 160
234 REM
235 REM AVION ATERRIZADO O ESTRELLADO?
236 REM
239 REM SE HA BATIDO EL RECORD?
240 IF H>R THEN R=H
250 PRINT@3,"PUNTUACION :";H,
260 PRINT@35,"RECORD :";R;
270 FOR I=1 TO 100
280 NEXT I
290 R$=INKEY$
300 PRINT@73,"OTRA (S/N) ?";
310 R$=INKEY$
320 IF R$="" THEN 310
330 IF R$<>"N" THEN 40
340 CLS
350 END
354 REM
355 REM BOMBA LANZADA
356 REM
359 REM HA LLEGADO AL SUELO
360 IF B>=510 THEN B=0
365 REM COLOCACION DE LA BOMBA
370 PRINT@B1,CHR$(128);
380 IF B<>0 THEN
    PRINT@B,B$;:B1=B:B=B+32
390 RETURN
394 REM
395 REM RETRASO PARA DISMINUIR
396 REM LA VELOCIDAD DEL AVION
397 REM SI NO SE HA LANZADO BOMBA
398 REM
400 FOR I=1 TO 20
420 RETURN
```

Porción de memoria

El Wafadrive Rotronics constituye una interesante alternativa de almacenamiento masivo para el Sinclair Spectrum

A pesar de que el Sinclair Spectrum se ha convertido en uno de los microordenadores personales más populares, ha sido objeto de muchas críticas en el sentido de que es una máquina inadecuada para el usuario "serio". Ello ha llevado a que se le considere sólo como una máquina para juegos. Parte del problema se ha centrado en el teclado, que no ha permitido que el usuario considere seriamente utilizar la máquina para aplicaciones tales como proceso de textos y administración de bases de datos. Recientemente, Sinclair Research ha intentado acallar algunas de estas críticas mediante el lanzamiento del Spectrum+, que viene equipado con un teclado estilo QL.

Sin embargo, esto no es más que una parte del problema. Otras de las dificultades que acosan a los que consideran al Spectrum como una máquina seria, son la falta de interfaces estándares del ordenador y, lo que es aún más importante, de un sistema de almacenamiento masivo fiable y rápido, algo vital para cualquier aplicación seria o de gestión que desee el aficionado. Por supuesto, con la introducción de la Interface 1 y el microdrive, el Spectrum estuvo al menos en condiciones de hacer disponibles estas facilidades. Pero los recelos no se disiparon, pues se consideró que el microdrive era lento y poco fiable. Además, si bien la base de software en cassette para el Spectrum es inmensa, es muy poco el software que se ha adaptado al medio que utiliza el microdrive. En una situación como ésta, los proveedores independientes tienden a cubrir el vacío y producir alternativas. Aquí vamos a analizar al primero de dos competidores en la carrera por dominar el mercado de almacenamiento masivo para el Spectrum: el Wafadrive de Rotronics. En el próximo capítulo nos concentraremos en el Discovery 1 de Opus Supplies.

A diferencia de la Interface 1 y los microdrives que lo acompañan, el Wafadrive de Rotronics es una unidad de "todo en uno". Ello significa que tanto las interfaces para periféricos como las unidades para almacenamiento masivo se hallan en una única caja. La ventaja de este sistema reside en que las unidades no poseen los cables de ampliación necesarios para el sistema Sinclair; no obstante, carecen de parte de la flexibilidad de los microdrives, que se pueden encadenar fácilmente en margarita para ampliar el espacio de almacenamiento.

El aspecto de la máquina

El Wafadrive se halla alojado en una carcasa de plástico negro, con un cable plano de 35 vías que termina en una ranura para cartuchos instalada en el bus de ampliación del Spectrum. En la parte frontal de la unidad hay un par de ranuras para activar cartuchos *wafer*. Entre las ranuras hay tres diodos emisores de luz (LED). La luz del centro es el indicador de potencia, mientras que las otras dos indican la actividad de las unidades de disco.

En la parte posterior de la unidad hay tres conectores marginales. A la izquierda hay un bus de ampliación en paralelo que permite la conexión de la Interface 2. El conector marginal del centro es una interface compatible con Centronics que permite acoplarle a la unidad una impresora en paralelo. El tercero es una puerta en serie RS232 que permite la conexión en interface del dispositivo con modems y otros dispositivos en serie. Estas interfaces constituyen una mejora respecto a las que proporciona la Interface 1 de Sinclair, en la cual, por ejemplo, es necesario conectar una segunda interface Centronics al conector marginal de la unidad para poder operar impresoras en paralelo. Lamentablemente, sin embargo, los usuarios todavía tendrán que ir de tienda en tienda para conseguir impresoras Centronics o modems con conectores para cartuchos que sean compatibles con Wafadrive.

Los *wafers* (galletas) flexibles diseñados específicamente para el Wafadrive son en muchos sentidos similares a los utilizados en sus equivalentes de Sinclair. Dentro de cada *wafer* hay un bucle continuo de cinta de cassette tipo video, cuya anchura es de 1,8 mm. Se emplea esta cinta en vez de la de audio convencional debido a su mayor resistencia y capacidad de almacenamiento de información. Una vez formateada, esta cinta puede contener aproximadamente 128 K de datos, si bien Rotronics ha puesto a la venta también cartuchos de 64 y 16 K.

Los cartuchos son aproximadamente el doble de

Chris Stevens

Espacio posterior
A diferencia del propio sistema de almacenamiento masivo de Sinclair para el Spectrum, el Wafadrive de Rotronics viene en una única caja que contiene 2 unidades gemelas, una puerta RS232 y una interface Centronics. El cable plano se inserta en el conector marginal del Spectrum, con lo que las unidades se colocan convenientemente detrás del teclado, para un fácil acceso

anchos que los de Sinclair, aunque sus cajas protectoras tienen un largo y ancho similar. Ello les confiere a los cartuchos Rotronics una apariencia de cassettes en miniatura. Los cartuchos Wafadrive no necesitan carcasas protectoras, dado que la delicada cinta está protegida por una cubierta deslizante automática, de diseño similar a la de los microdiscos flexibles Sony de 3 ½ pulgadas, si bien la protección Rotronics es de plástico en vez de metal. En el lado izquierdo del *wafer* hay una traba de protección de escritura, que se puede quitar. Por supuesto, si esta traba se rompe ya no se podrá reemplazar, por lo que los usuarios habrán de pensar en otro método para poder utilizar sus cartuchos.

Las instrucciones que utiliza el Wafadrive son similares a las que emplean los microdrives de Sinclair. En ambos sistemas, la instrucción va seguida de un *, que indica que se ha de acceder al dispositivo de almacenamiento externo. Ejemplos de esta forma de empleo son SAVE*, LOAD* y VERIFY*. No obstante, el sistema Wafadrive posee ligeras diferencias, porque siempre hay presentes dos Wafadrives, frente a los numerosos microdrives que podrían incluirse en el sistema. Por ejemplo, cuando se formatea el microdrive uno de Sinclair, se utiliza la instrucción FORMAT 'm';0;"nombre", donde 'm';0 alude al número del microdrive que se está empleando. Cuando se utiliza el Wafadrive, la instrucción es FORMAT*'a:nombre', aludiendo la a: al nombre de la unidad de disco en uso. Observe que con el Wafadrive sólo puede haber una sección a: o b:, mientras que en el microdrive el número puede ser del cero al siete.

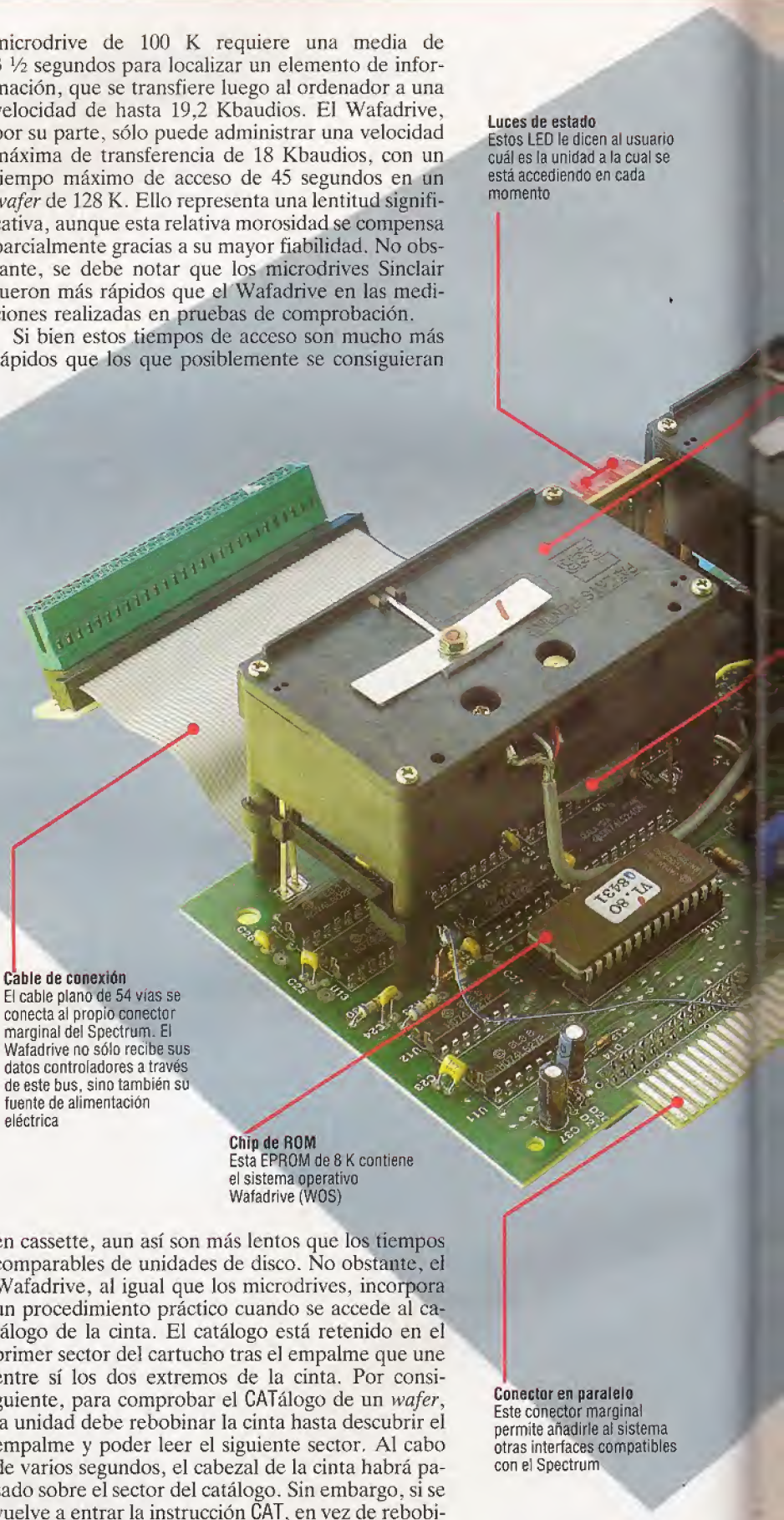
El sistema de "corrientes"

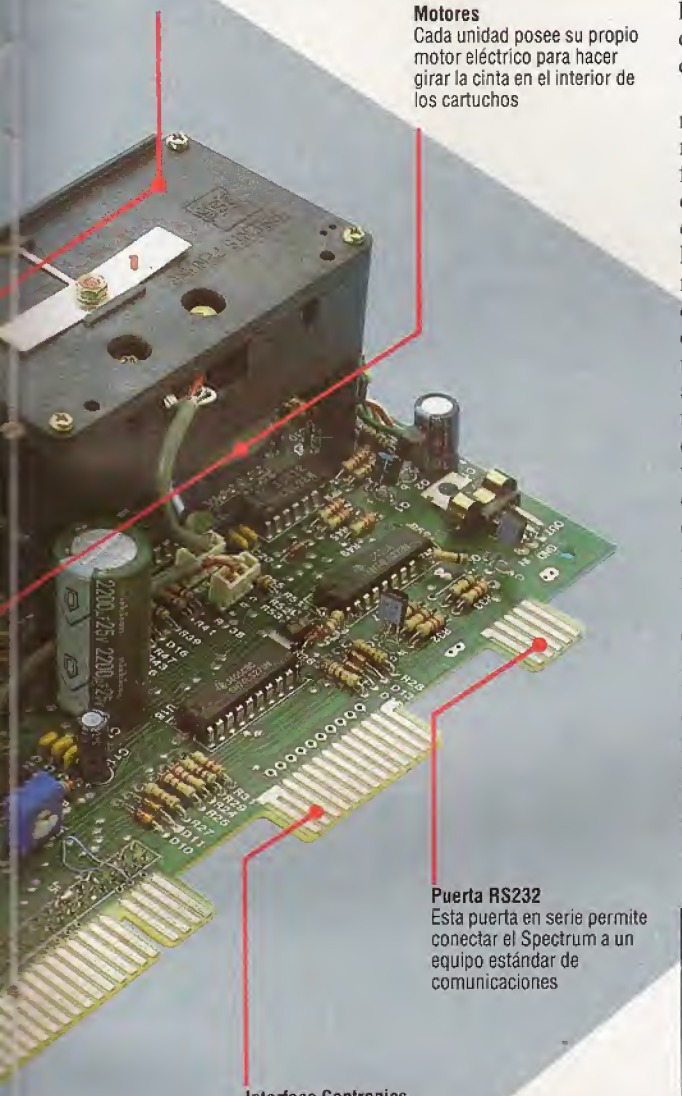
El Wafadrive también saca provecho del sistema de "corrientes" (*streams*) utilizado en el Spectrum, en el cual hay 16 corrientes reservadas para el tratamiento de E/S; algunas de éstas están reservadas para uso de la pantalla y la impresora. Sin embargo, los canales del cuatro al quince están disponibles para otros dispositivos periféricos, y a las corrientes de salida al Wafadrive se accede mediante el empleo de la instrucción OPEN#. Asimismo, el Wafadrive añade al sistema dos corriente adicionales. Los canales *r* y *c* (estas letras también pueden ir en mayúscula) están reservados para las interfaces RS232 y Centronics, respectivamente, y su empleo es similar al de los canales *t* y *b* (utilizados cuando se accede a la puerta RS232 en la Interface 1).

A bordo hay una ROM de 8 K que contiene las instrucciones del BASIC extendido que se utilizan para controlar el sistema. Este sistema operativo Wafadrive (WOS: Wafadrive Operating System) es capaz de funcionar paginando los 8 K inferiores de la ROM del Spectrum, de forma muy similar a la forma en que lo hace la propia Interface 1. Por ejemplo, la instrucción LOAD* en realidad genera un error en el Spectrum; por consiguiente, cuando el intérprete de BASIC encuentre esta instrucción en la pantalla, llamará a la rutina para tratamiento de errores. No obstante, el WOS puede interceptar esta instrucción de llamada, paginándola entonces a la ROM del Wafadrive. Ésta, a su vez, asumirá el tratamiento de errores e interpretará LOAD* como una instrucción.

En comparación con los microdrives, el Wafadrive de Rotronics es más bien lento. Por ejemplo, un microdrive de 100 K requiere una media de 3 ½ segundos para localizar un elemento de información, que se transfiere luego al ordenador a una velocidad de hasta 19,2 Kbaudios. El Wafadrive, por su parte, sólo puede administrar una velocidad máxima de transferencia de 18 Kbaudios, con un tiempo máximo de acceso de 45 segundos en un *wafer* de 128 K. Ello representa una lentitud significativa, aunque esta relativa morosidad se compensa parcialmente gracias a su mayor fiabilidad. No obstante, se debe notar que los microdrives Sinclair fueron más rápidos que el Wafadrive en las mediciones realizadas en pruebas de comprobación.

Si bien estos tiempos de acceso son mucho más rápidos que los que posiblemente se consiguieran en cassette, aun así son más lentos que los tiempos comparables de unidades de disco. No obstante, el Wafadrive, al igual que los microdrives, incorpora un procedimiento práctico cuando se accede al catálogo de la cinta. El catálogo está retenido en el primer sector del cartucho tras el empalme que une entre sí los dos extremos de la cinta. Por consiguiente, para comprobar el CATálogo de un *wafer*, la unidad debe rebobinar la cinta hasta descubrir el empalme y poder leer el siguiente sector. Al cabo de varios segundos, el cabezal de la cinta habrá pasado sobre el sector del catálogo. Sin embargo, si se vuelve a entrar la instrucción CAT, en vez de rebobinar otra vez toda la cinta la unidad vuelve a visualizar otra vez el catálogo tras una fracción de segundo: el mismo queda retenido en RAM una vez que





Unidades de "wafer"
El Wafadrive posee dos unidades de cinta, ambas autocontenidas. Rotronics ofrece para instalar en ellas cartuchos de 16, 64 y 128 K

Motores
Cada unidad posee su propio motor eléctrico para hacer girar la cinta en el interior de los cartuchos

Puerta RS232
Esta puerta en serie permite conectar el Spectrum a un equipo estándar de comunicaciones

Interface Centronics
El Wafadrive está equipado con una puerta Centronics que permite la adición de una impresora, necesaria para tratamiento de textos

se lo llama. A partir de entonces, el WOS se limita a comprobar si el *wafer* insertado es el mismo, examinando el siguiente sector. De ser así, el WOS visualiza el catálogo que ya tiene retenido en RAM.

Manteniéndose fiel a la idea de implementar un sistema de almacenamiento masivo que permita utilizar el Spectrum para aplicaciones más serias, Rotronics ha incluido en el paquete el programa para tratamiento de textos Spectral Writer. Se trata de un sistema relativamente amplio que hace un uso cabal del Wafadrive. A las funciones tales como reformatear un párrafo, insertar palabras y suprimir líneas se las llama utilizando la tecla Symbol Shift del Spectrum junto con otras teclas. Otras funciones que acceden a archivos retenidos en *wafer* se pueden obtener mediante la instrucción de opciones: éstas incluyen guardar (SAVE) y cargar (LOAD) archivos de texto desde cassette o desde el Wafadrive. Spectral Writer es un procesador de textos atractivo, si bien no permite que uno establezca la longitud de la línea en pantalla. Quizá sea lamentable que aún utilizando un Spectrum+, la calidad del teclado desluzca en cierto modo la eficacia del Spectral Writer.

Por supuesto, el factor de mayor peso que determina el éxito o el fracaso de todo medio de almacenamiento para ordenador es la disposición de las firmas de software para apoyarlo. Por el momento éste parece ser un grave inconveniente para el éxito del Wafadrive, puesto que ninguna de las principales firmas de software está produciendo sus programas en cartuchos Wafadrive (éste es un problema que también ha afectado a la propia Sinclair). No obstante, no todo está perdido para el usuario de un Wafadrive. Hay al menos una empresa que en la actualidad está produciendo un programa que permite que uno vuelque software comercial en cartuchos Wafadrive. Esto significa que los usuarios se verán obligados a adquirir tanto la cassette comercial como un *wafer* para efectuar en él la transferencia, pero podría ser un bajo precio pagar para obtener unos tiempos de acceso considerablemente mejores.

Otra pequeña dificultad del Wafadrive son los conectores marginales de la parte posterior de la máquina. Debido a que éstos no son estándares, los usuarios habrán de convertir las propias interfaces, o bien esforzarse por encontrar periféricos con conexiones apropiadas. Pero aunque el Wafadrive posee sus inconvenientes, es una máquina muy bien hecha y, por cierto, representa una alternativa viable a la Interface 1 y microdrive que suministra Sinclair Research.

WAFADRIVE ROTRONICS

DIMENSIONES
230×110×80 mm

INTERFACES
Puerta en serie RS232, interfaces en paralelo Centronics, conector marginal del Spectrum

FORMATO
Wafers flexibles de bucle continuo

CAPACIDAD
Wafers de 16, 64 y 128 K

VELOCIDAD
Velocidad de transferencia: 16 Kbaudios; tiempo máximo de acceso: 6,5 seg (16 K), 45 seg (128 K)

Cartucho Wafadrive
Desde el punto de vista de las proporciones, el *wafer* se asemeja a una cassette convencional, pero la cinta que hay en su interior está empalmada formando un bucle continuo. Esto significa que no se ha de rebobinar la cinta para acceder a los datos que ya han pasado por el cabezal de lectura/escritura

Chris Stevens

En alta mar

Ahora que ya hemos levado anclas, debemos llevar un recuento semanal de la tripulación y el barco

Durante el viaje al Nuevo Mundo, la velocidad del barco se ve afectada por el número y fortaleza de los tripulantes. Una tripulación numerosa y saludable no contribuye a aumentar la velocidad del barco, pero cuando la tasa de fortaleza combinada disminuye por debajo de cierto punto, la ineficacia incrementa la duración del viaje. Si toda la tripulación muere, cómo es lógico, el juego se termina.

Una de las funciones del Diario de Navegación del Capitán es informar al jugador sobre la muerte de cualquier miembro de la tripulación, si se ha agotado alguna provisión, y visualizar un cálculo actualizado de la duración de lo que aún resta del viaje. En la línea 41 se crea una nueva variable, EW. Ésta representa el tiempo extra del viaje expresado en semanas (o fracciones de semana), que se sumará a las ocho semanas originales si la fortaleza acumulada de la tripulación cae por debajo de un cierto nivel. Para que la duración del viaje se pueda visualizar e incrementar en enteros sin perder la parte fraccional de EW, el tiempo extra para el viaje se calcula y se almacena en EW como un número real, pero se le suma a JL como un entero, utilizando INT.

La programación para el informe de final de semana se halla en la subrutina de la línea 5300. Habrá una gran sección del programa para tratar con las contingencias que se produzcan durante el viaje, que se irán añadiendo en futuros módulos y que comenzaremos a programar en el próximo capítulo. Estos acontecimientos incidirán también en la cantidad de provisiones, la fortaleza y el número de tripulantes y la duración del viaje y, por consiguiente, influirán en los valores impresos en la sección de este capítulo.

La primera tarea de la subrutina del informe de final de semana consiste en establecer un bucle en la línea 5325 para comprobar si alguno de los tripulantes ha fallecido. Examina el segundo elemento de la matriz de tripulación, TS(,), para cada uno de los 16 posibles tripulantes, y determina si la tasa de fortaleza está o no establecida en −999, lo que determina la muerte.

Los fallecidos se le suman a X, que es el contador semanal de las víctimas. Si el valor de la matriz no es igual a −999, se implementa el siguiente contador del bucle. Tras registrar una pérdida, la línea 5340 restablece a cero los elementos de fortaleza y categoría, para evitar que la muerte se vuelva a registrar en el siguiente informe de final de semana. Se resta entonces el marinero del total de la tripulación CN. Si no ha habido fallecimientos, el programa va a la línea 5400, donde se comprueban las provisiones. Si toda la tripulación ha muerto, el programa imprime "Toda la tripulación que quedaba ha fallecido. Su barco navega a la deriva a través del océano desierto. El juego ha terminado." Si viven todavía algunos tripulantes entonces en la línea 5390 se imprime el número de fallecidos.

Las provisiones consumidas se registran de la misma forma que los tripulantes que han perdido la vida. El programa comprueba si se ha agotado alguna provisión y prepara un bucle de 1 a 4 para representar los 4 tipos de provisiones. Si se ha agotado alguna, su valor en la matriz PA() será −999. Preparada durante la rutina "reparto de provisiones", la línea 5415 comprueba PA() e informa al jugador que "Te has quedado sin..." ese elemento en particular si en realidad el valor es de −999.

El elemento de PA() se restablece luego a 0, para impedir que al finalizar la siguiente semana se vuelva a imprimir el mensaje. Si se ha agotado más de una clase de provisiones, antes de imprimir su nombre se imprime "Y". Es posible que alguna de las provisiones agotadas se pueda conseguir en la sección de contingencias del juego. De ser así, se restablecerá el elemento de PA() para esa provisión, y figurará en todos los registros como normal.

El tiempo adicional para el viaje se calcula revisando la matriz de fortaleza de la tripulación, acumulando la fortaleza total y comparándola con la fortaleza óptima mínima. Para continuar el viaje con una eficiencia máxima, la fortaleza total de la tripulación ha de ser de al menos 800. Ello se puede conseguir con ocho tripulantes con todas sus fuerzas, o con 16 con una fortaleza media, y así sucesivamente. La línea 5465 prepara un bucle para cada elemento de la matriz de la tripulación y la 5470 acumula las fortalezas en X.

La línea 5480 comprueba si la fortaleza total es mayor que 799. De ser así, no aumentará el tiempo de duración del viaje; el programa pasa entonces a la línea 5494, que solicita que se pulse una tecla para dar comienzo a la siguiente semana.

Si la fortaleza es inferior a 800, la fórmula de la línea 5489 calcula el tiempo extra para el viaje. Le resta a 800 la fortaleza acumulada de la tripulación, divide por 800 y le suma el resultado al tiempo extra del viaje. El incremento se almacena en EW como un número real. La línea 5490 le suma la parte entera de EW a la duración del viaje, para que el tiempo de viaje restante no se visualice con fracciones de semana. El valor acumulado de EW se suma cada semana a la duración del viaje. Para que la tasa de fortaleza de un tripulante fallecido no disminuya en función de WF en la subrutina de la línea 9300 que ya hemos ofrecido, inserte esta línea:

```
9315 IF TS(S1,2)=0 THEN 9340
```

Mientras el barco vaya navegando hacia el Nuevo Mundo, varios acontecimientos influirán en la fortaleza de la tripulación. Hasta ahora, el único factor del programa que afectaba a la fortaleza era la

Mód. 5: Diario de navegación

Informe de final de semana

```
5300 REM INFORME DE FINAL DE SEMANA
5305 PRINTCHR$(147)
5310 S$="    DIARIO DE NAVEGACION*":GOSUB 9100
5312 S$="    ------ -- ----------*":GOSUB 9100
5314 GOSUB 9200
5316 PRINT:PRINT"          FIN DE LA SEMANA";WK
5318 GOSUB 9200
5320 X=0
5325 FOR T=1 TO 16
5330 IF TS(T,2)<>-999 THEN 5350
5335 X=X+1
5340 TS(T,1)=0:TS(T,2)=0
5345 CN=CN-1
5350 NEXT
5355 IF X=0 THEN 5400
5358 PRINT:PRINT
5360 S$="DURANTE LA SEMANA PASADA*":GOSUB 9100
5365 IF CN>0 THEN 5390
5367 S$="MURIO TODA LA TRIPULACION QUE
     QUEDABA*":GOSUB 9100
5369 GOSUB 9200:PRINT:GOSUB 9200
5375 S$="EL BARCO VA IRREMISIBLEMENTE A LA
     DERIVA*":GOSUB 9100
5374 GOSUB 9200
5376 S$="POR EL OCEANO DESIERTO...*":GOSUB 9100
5378 GOSUB 9200
5380 PRINT:PRINT
5382 S$="      EL JUEGO HA TERMINADO*":GOSUB 9100
5384 PRINT:PRINT
5386 GOSUB 9200:END
5388 GOTO 5386
5390 PRINT X;
5392 IF X=1 THEN S$="TRIPULANTE HA FALLECIDO*"
5394 IF X<>1 THEN S$="TRIPULANTES HAN FALLECIDO*"
5396 GOSUB 9100
5398 GOSUB 9200
5400 PRINT:PRINT
5405 S$="AHORA TE HAS QUEDADO SIN "
5410 FOR T=1 TO 4
5415 IF PA(T)<>-999 THEN 5440
5420 PRINT S$;P$(T)
5425 PA(T)=0
5428 S$="                    Y"
5440 NEXT
5450 GOSUB 9200
5455 REM CALCULAR NUEVA JL
5460 X=0
5465 FOR T=1 TO 16
5470 X=X+TS(T,2)
5475 NEXT
5480 IF X>799 THEN 5494
5481 PRINT:PRINT
5482 S$="LA TRIPULACION SE HALLA POR DEBAJO DE LA
     FORTALEZA TOTAL*":GOSUB 9100
5483 S$="DE MANERA QUE EL VIAJE PODRIA DURAR
     MAS*":GOSUB 9100
5489 EW=EW+((800-X)/800)
5490 JL=JL+INT(EW)
5492 PRINT
5494 S$=KS:GOSUB 9100
5496 GET I$:IF I$="" THEN 5496
5499 RETURN
```

Variable de las semanas extras

```
41 EW=0:REM SEMANAS EXTRAS
```

Adición al bucle principal del viaje

```
880 GOSUB 5300:REM INFORME DE FINAL DE SEMANA
```

Complementos al BASIC

Spectrum:

Introduzca las siguientes modificaciones:

```
5305 CLS
5496 LET I$=INKEY$:IF I$="" THEN GOTO
     5496
```

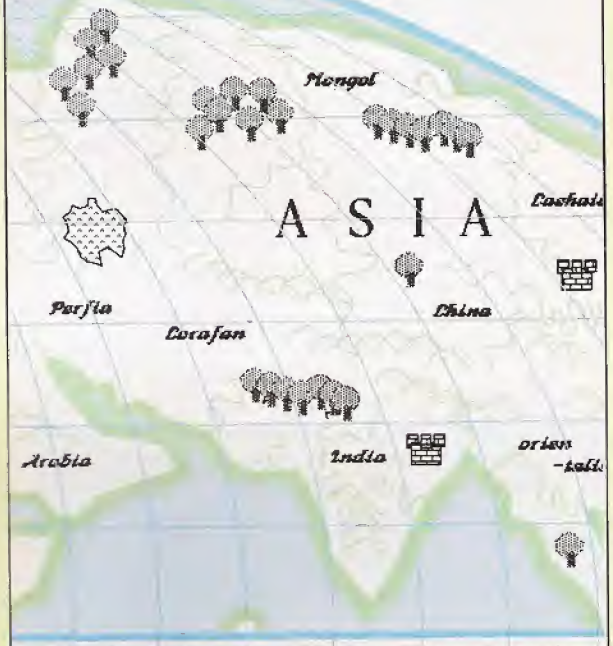
BBC Micro:

Introduzca estos cambios:

```
5305 CLS
5496 I$=GET$
```

Ian McKinnell

La travesía hasta ahora
Después de añadir el Módulo 5 del programa, podemos repasar las etapas que ya hemos programado en nuestro juego de simulación Nuevo Mundo, valiéndonos de un diagrama de flujo. La estructura del programa es lineal durante las etapas iniciales de pedidos y contratación, pero hace uso de un bucle simple para contar las semanas del viaje



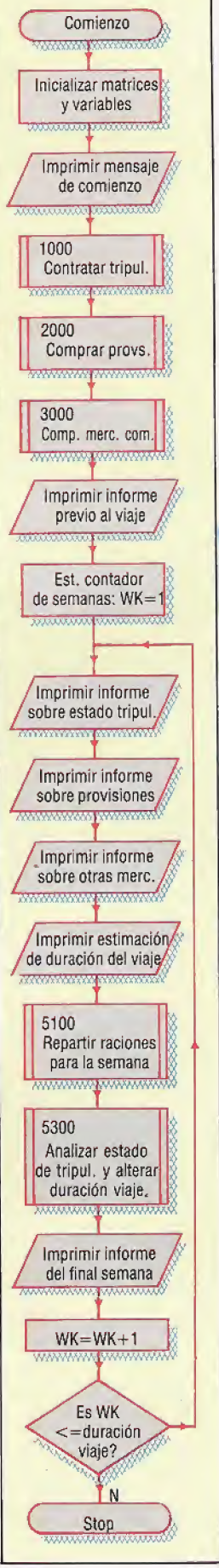
dieta. Dado que la comida se comparte en porciones iguales, una dieta inadecuada reduce la fortaleza de todos los miembros de la tripulación en la misma proporción. Para observar las consecuencias de reducir las tasas de fortaleza individuales en esta etapa del juego, podemos insertar una pequeña rutina que establezca las tasas de fortaleza al azar al comienzo del viaje. Sin embargo, si utiliza esta rutina, deberá acordarse de suprimirla antes de entrar la próxima entrega del programa.

```
601 REM CODIGO DE PRUEBA
602 FOR T=1 TO 16
603 IF TS(T,2)=100 THEN TS(T,2)=
    INT(RND(1)*100)+1
604 NEXT
```

La rutina de la línea 603 revisa el segundo elemento de la matriz de la tripulación, en 16 iteraciones. Cambia toda tasa de fortaleza de 100 por un número elegido al azar entre 0 y 99, utilizando la fórmula RND dada (a cada número al azar se le suma 1 para evitar que se establezca una tasa de fortaleza en cero, lo que determinaría la muerte del tripulante).

La rutina del informe de final de semana se llama mediante una llamada GOSUB desde dentro del bucle principal del viaje, en las líneas 820-889. Originalmente, este bucle estaba controlado por una construcción FOR...NEXT, estableciendo el límite máximo la variable de la duración del viaje, JL. Sin embargo, dado que el nuevo módulo puede alterar el valor de JL, necesitamos reconstruir el bucle reemplazando las sentencias FOR y NEXT por:

```
820 WK=1:REM BUCLE PRINCIPAL DEL VIAJE
889 WK=WK+1:IF WK<=JL THEN 825
```



Registros

En PASCAL, los registros representan la forma más útil de empaquetar juntos datos de distinto tipo

En el capítulo anterior, le mostramos cómo los conjuntos pueden ser sumamente útiles para el programador de PASCAL, y que se pueden implementar con eficacia a nivel máquina porque las operaciones definidas para ellos poseen equivalentes directos en los juegos de instrucciones nativos de la mayoría de los procesadores. Siempre se construyen reuniendo un grupo de elementos que pueden ser de cualquier tipo escalar (no real). Sin embargo, aparte de poderse comprobar la pertenencia mediante el operador IN, todas las operaciones de conjuntos están definidas sobre la totalidad de la estructura. Si bien no hay ningún mecanismo de selección para extraer un elemento en particular, podemos utilizar los operadores de inclusión de conjunto (<=>=) y comprobar la equivalencia de conjuntos ("igualdad") con = y <>.

Cuando los conjuntos los manipulan matemáticos, los mismos no sólo pueden contener objetos ilimitados, sino que también se los puede considerar teóricamente infinitos (tales son las ventajas de resolver problemas en un trozo de papel, respecto a un hardware físicamente limitado). La implementación que usted posee pondrá algún límite tanto al tamaño como a la gama de los conjuntos, de modo que no intente definir tipos ambiciosos como:

```
GranConj=SET OF enteros;
```

El límite inferior de un elemento perteneciente a un conjunto debe tener un valor ordinal de cero o mayor, y el límite superior estará restringido a algún valor absoluto: entre 255 y 4 095 en implementaciones de microordenador. Observe que el conjunto vacío (representado literalmente mediante []) es miembro de todos los conjuntos posibles, sea cual sea su tipo. Esto podría parecer un agujero en la sólida armadura tipológica del PASCAL, pero, en la práctica, el tipo de conjunto vacío siempre se puede deducir a partir del tipo de otros conjuntos de cualquier expresión.

La inclusión estricta de subconjuntos verdaderos no se puede obtener directamente, y los operadores < y > no están definidos para estructuras de conjunto. Ello se debe a razones de implementación: la mayoría de las restricciones del PASCAL tienen sentido desde el punto de vista de la eficacia o la lógica. Si usted deseara comprobar una inclusión estricta, sería necesaria una comparación doble, por ejemplo:

```
(A>=B) AND (A<>B)
```

La prueba para House en nuestro programa *Bingo* no exigía esta comparación especial, puesto que un conjunto de números cantados puede que sea equivalente ("igual") o bien, lo que es más probable, un subconjunto de los números del cartón. De modo que, por ejemplo, esta expresión booleana se podría haber expresado como:

```
House :=Cantando>=Carton
```

que resulta verdadera cuando todos los miembros de Carton están incluidos en el conjunto de los números Cantado. Ésta es exactamente la clase de propiedad fundamental de los conjuntos que los convierte en valiosísimas estructuras de datos para la resolución de problemas. Por ahora, probablemente la aplicación más útil de los conjuntos será para la comprobación de subconjuntos del tipo char. El siguiente esquema (en PASCAL/castellano) sería de utilidad para programar cualquier aplicación interactiva, como por ejemplo un juego.

```
Negativo :=['N','n'];
Afirmativo :=['S','s'];

REPEAT
   {Desarrollar el juego}
   {visualizar el(los)marcador(es)}
   write ('Otra vez?');
   ReadLn (respuesta);

   WHILE NOT (respuesta IN Afirmativo+
   Negativo)DO
      BEGIN
         WriteLn('S(i) o N(o)':Columnas);
         write ('Otra partida?');
         ReadLn (respuesta)
      END

UNTIL respuesta IN Negativo
```

Cuando deseamos acceder a elementos individuales de una estructura, las alternativas de que disponemos son matrices (suficientemente familiares), archivos y registros. Estos últimos poseen la eficaz habilidad de reproducir registros de datos de la vida real, con "campos" mezclados de cualquier tipo de datos: simples o estructurados.

La forma de registro de datos más común que se utiliza en el campo de gestión contiene campos para nombres, direcciones, números de teléfono, códigos de cuentas, etc. Lo importante es poder manipular estos datos como un bloque de información y *también* poder acceder a cualquier campo individual y procesar adecuadamente los datos. El PASCAL permite asignar tales objetos (y manipularlos de otras maneras) como un todo, permitiendo, al mismo tiempo, acceder a cualquiera de los campos componentes con fines de comparación o proceso, en función de su tipo de datos individual. La definición de un registro mixto es bastante directa:

```
TYPE
   habitacion=RECORD
            numero  : 1..999;
            ala     : (Norte,Este,Sur,Oeste);
            ocupada : boolean
      END;{habitacion}
VAR
   servicio:habitacion;
```

Así como la sentencia CASE era excepcional en el uso de la palabra reservada END como delimitador,

la definición de un registro constituye la única excepción, en la parte de declaración de un bloque, a la regla de que los BEGIN y END se utilizan de a pares. Por este motivo resulta útil complementar el END de un registro con su identificador, como en nuestro ejemplo. Cualquier variable de tipo habitacion contendrá tres campos componentes. En este caso, cada campo es de un tipo escalar distinto, pero podrían igualmente ser del mismo tipo, simple o estructurado. No existe restricción en cuanto a los tipos permitidos dentro de los campos de un registro, ¡de modo que podemos tener un campo que sea una matriz de archivos de registros que contengan conjuntos!

Dentro de las palabras delimitadoras RECORD y END, la sintaxis de definición es exactamente igual que en una declaración VAR. Aquí, sin embargo, estamos declarando identificadores de campo que son una parte integral de la estructura del registro. Por ello los nombres numero, ala y ocupada no existen fuera del "ámbito" del identificador del registro. Estos nombres de campo son identificadores locales y podrían duplicar los nombres de variables del programa. Sólo se puede acceder a ellos a través de dos mecanismos que posee el PASCAL para seleccionar registros: la notación de "punto" y la sentencia WITH.

Para seleccionar un campo determinado utilizando la notación de punto, se separa el identificador de todo el registro del identificador de campo siguiente mediante un punto ("."). Por ejemplo:

```
servicio.numero
```

aludiría solamente al campo del subrango de enteros. Podríamos inicializar el contenido del registro con sentencias tales como:

```
read (servicio.numero);
'servicio.ala:=Este;'
'servicio.ocupada:=persona<>[]'
```

etcétera. Observe que nos hemos introducido en un conjunto vacío con un tipo que depende del tipo, cualquiera que sea, de persona, ¡y ni siquiera lo hemos verdaderamente declarado!

La sentencia WITH

La notación de punto se puede volver algo engorrosa en los casos en que deseemos acceder a la mayoría o la totalidad de los campos de un registro. Existe una sentencia estructurada alternativa que "descubre" los identificadores de campo. La semántica de la sentencia WITH significa, en líneas generales: "Quiero hacer algo con este registro, de modo que deseo especificar sólo los nombres de los campos." La sintaxis de la sentencia WITH tiene exactamente la misma forma que el bucle WHILE, y la secuencia de asignaciones de inicialización que hemos dado se podría expresar mejor de este modo:

```
WITH servicio DO
  BEGIN
    numero:=123;
    ala:=Este;
    ocupada:=true
  END
```

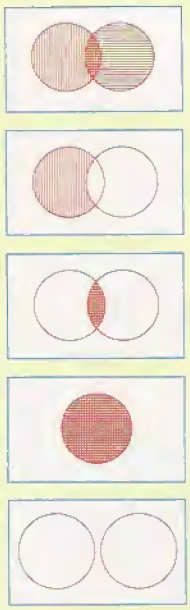
A lo largo de la sentencia WITH (desde el BEGIN al END, en este caso), no se han de cualificar los identificadores de campo con el identificador del registro y el punto, y se puede aludir a ellos directamente, como podemos ver. Aquí no habría ninguna confusión si hubiera otra variable del programa denominada Numero, por ejemplo. El ámbito local siempre tiene precedencia. Sin embargo, utilizando la notación de punto podemos comunicar valores a través de estos límites del ámbito. A modo de ejemplo, tomemos:

```
servicio.numero:=Numero
```

Operaciones de conjuntos

Ofrecemos una lista de todas las operaciones sobre conjuntos disponibles en PASCAL (con diagramas de Venn en los casos apropiados)

1. *Unión* (C1+C2): Es el superconjunto que comprende a todos los objetos pertenecientes tanto al conj. C1 como al conj. C2 combinados (diag. 1)
2. *Diferencia* (C1−C2): El subconjunto de los miembros de C1 que no pertenecen a C2 (diag. 2)
3. *Intersección* (C1*C2): El subconjunto de los objetos comunes a ambos conjuntos (diag. 3)
4. *Equivalencia* (C1=C2): Donde los miembros pertenecientes a C1 y C2 son idénticos (diag. 4)
5. *No equivalencia* (C1<>C2): Es verdadera cuando ninguno de los miembros de C1 es miembro de C2, y viceversa (diagrama 5)
6. *Inclusión* (C1<=C2): Es verdadera si todos los miembros de C1 son también miembros de C2
7. *Inclusión* (C1>=C2): Es verdadera si todos los miembros de C2 son también miembros de C1
8. *Pertenencia* (m IN C1): Es verdadera si el miembro de un conjunto de un solo elemento ([m]) es un subconjunto de C1

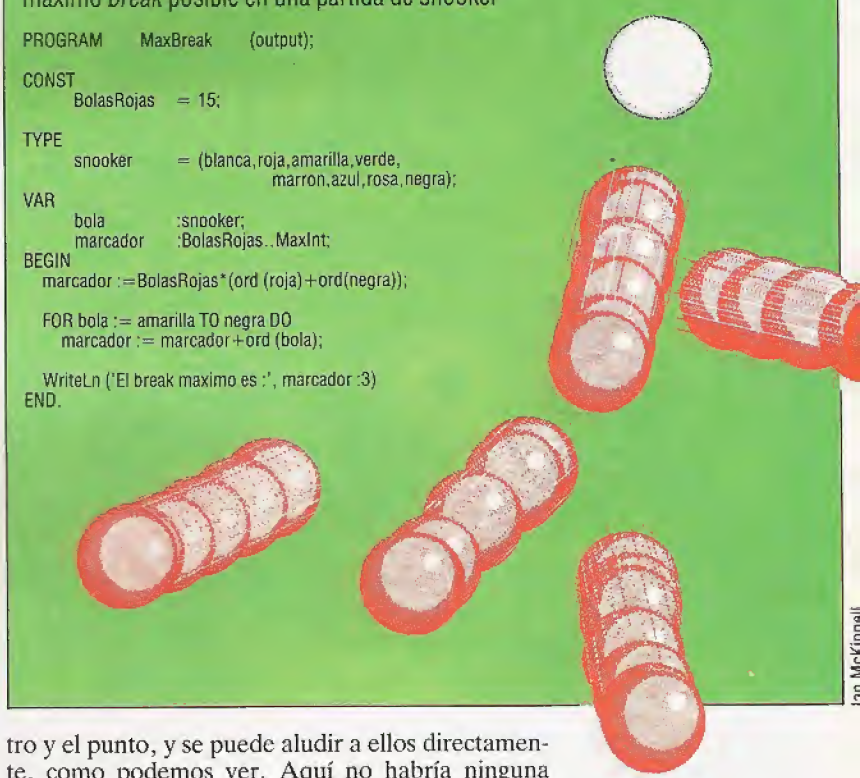


Liz Dixon

El máximo "break"

Ésta es nuestra solución al problema que planteamos en el capítulo anterior, en el que le pedíamos que creara un programa que calculara el máximo *break* posible en una partida de snooker

```
PROGRAM     MaxBreak     (output);

CONST
      BolasRojas    = 15;

TYPE
      snooker       = (blanca,roja,amarilla,verde,
                       marron,azul,rosa,negra);
VAR
      bola          :snooker;
      marcador      :BolasRojas..MaxInt;
BEGIN
  marcador :=BolasRojas*(ord (roja)+ord(negra));

  FOR bola := amarilla TO negra DO
    marcador := marcador+ord (bola);

  WriteLn ('El break maximo es :', marcador :3)
END.
```



Ian McKinnell

que le asignaría el valor de una variable externa (Numero) al campo del registro (numero). Recuerde que el PASCAL no hace distinciones entre tipos de letras (es decir, no hace ninguna diferencia entre letras en mayúscula y letras en minúscula), y el identificador no cualificado numero aludiría a una variable externa, a menos que se hallara dentro de una sentencia WITH. En todo caso, sería imprescindible utilizar la notación de punto si tuviéramos una asignación entre dos variables del mismo tipo de registro, supongamos:

```
grupos.ala:=recepcion.ala
```

Aquí la sentencia WITH sólo se podría emplear para descubrir uno de los campos de la variable, porque de lo contrario se produciría una confusa ambigüedad que no sería aceptada por el compilador.

La decisión más acertada que podríamos adoptar sería la siguiente:

```
WITH grupos DO
  ala:=recepcion.ala
```

En consecuencia, las dos formas de notación tienen su uso propio. La forma que se ha de utilizar resulta obvia a partir de la aplicación.

Un largo recorrido

El programa *Distancias* lee desde el teclado dos distancias expresadas en yardas, pies y pulgadas. Se utiliza una definición de tipo de registro para asignar un campo separado para cada valor de las unidades de medida diferentes. El programa suma luego las longitudes entre sí, pasando las pulgadas al campo "pies" y los pies de más al campo "yardas", mediante el empleo de los operadores de enteros DIV y MOD para dar la suma y el resto, respectivamente. Los resultados se asignan a los campos de "Total" y después se imprimen. La asignación de registros enteros sería directa (p. ej.: LongA:=LongB), pero todo proceso o manipulación se debe aplicar a cada uno de los campos componentes, de modo que Total:=LongA+LongB es ilegal. Observe el empleo de la notación de punto y la sentencia WITH para leer los valores de LongA y LongB.

```
PROGRAM     Distancias     (input, output);
CONST
        MaxByte     = 255;
TYPE
        byte        = 0..Maxbyte;
        distancia   = RECORD
                        rpulgadas   : 0..11;
                        rpies       : 0..2;
                        ryardas     : byte
                      END;{distancia}
VAR
        pulgadas,
        pies        : byte;
        LongA,
        LongB,
        Total       : distancia;
BEGIN
  WriteLn ('Entre las longitudes en pulgadas pies y yardas,');
  WriteLn ('separadas por SPACE o RETURN.');
  WriteLn;
  write ('Primera:':15);
  read (LongA.ryardas,LongA.rpies, LongA.rpulgadas);
  write ('Segunda:':15);
  WITH LongB DO
    read (ryardas,rpies,rpulgadas);
  pulgadas:=LongA.rpulgadas+LongB.rpulgadas;
  pies:=LongA.rpies+LongB.rpies+pulgadas DIV 12;
  WriteLn;
  WITH Total DO
    BEGIN
      rpulgadas:=pulgadas MOD 12;
      rpies:=pies MOD 3;
      ryardas:=LongA.ryardas+LongB.
               ryardas+pies DIV 3;
      WriteLn ('La longitud total es:');
      WriteLn (ryardas:25,'yardas',rpies:
               1,'piesy',rpulgadas:1,'pulgadas')
    END
END
```

Protección de datos

A partir de nuestro programa *Bingo* le habíamos propuesto a modo de ejercicio que le añadiera al programa construcciones de bucle para impedir la repetición de un número cantado y detectar cualquier número ilegal fuera de la escala entre 1 y 90. La mejor forma de proteger al programa de la entrada de datos equivocados consiste en añadir una construcción WHILE tras las sentencias ReadLn. La estructura será la misma en cada caso:

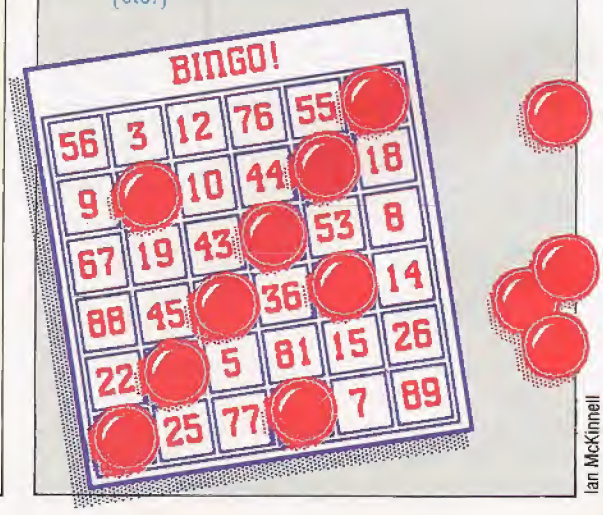
```
WHILE el número sea inaceptable DO
  dar un mensaje de error apropiado
  escribir otra petición
  volver a leer los datos
```

En el caso de las entradas iniciales para el cartón, debemos asegurarnos de que el número entrado no esté fuera de la escala permitida (de 1 a 90), y también evitar la repetición de una entrada previa. Por consiguiente:

```
WHILE NOT (numero IN Permitido)OR(numero
IN Carton)DO
  BEGIN
    WriteLn (numero:20,'no es valido');
    write ('Vuelva a entrar:':14);
    ReadLn (numero)
  END;
    {etc.}
```

Para cantar los números se aplica la misma estrategia, pero con la condición para entrar en el bucle de comprobación alterada:

```
WHILE NOT (numero IN Permitido)OR
(numero IN Cantado)DO
  {etc.}
```



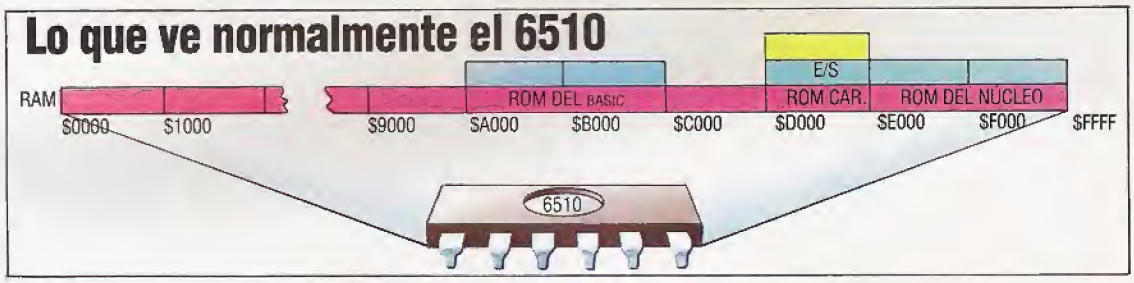
Ian McKinnell

La vuelta al bloque del Commodore 64

Iniciamos una serie de capítulos dedicados al estudio de las aplicaciones prácticas ofrecidas por el OS del Commodore 64

La primera cosa que necesitamos saber sobre el ordenador desde un punto de vista de programación de sistema es la estructuración general del mapa de la memoria. En el caso del Commodore 64 hay que decir que el microprocesador 6510 puede "ver" uno entre "varios" mapas, según cómo el programador configure la máquina. Esto se debe a que el 6510 es un microprocesador muy versátil con posibilidad de conmutar la conexión con diferentes bloques de memoria. Esto lo consigue el hardware poniendo las patillas 8 y 9 de la puerta de ampliación arriba (+5 V) o abajo (0 V), o bien, el software, alterando el contenido de la dirección 1. Cualquiera de estos dos procedimientos alterará radicalmente la manera en que el 6510 ve la memoria. Ofrecemos aquí una figura esquemática del mapa normal (por defecto), mostrando los 64 K de memoria que el 6510 ve normalmente. La tabla adjunta hace un detallado reconocimiento del mapa de la memoria.

Lo mismo que todo programa en BASIC necesita una RAM donde almacenar sus variables, así las rutinas del intérprete y del núcleo necesitan una RAM del OS. La RAM del OS se encuentra en las posiciones entre $0002 y $03FF, y uno de sus componentes importantes, la pila, ocupa las posiciones entre $0100 y $01FF. La RAM del OS se coloca en la página cero porque forma parte del mapa de la memoria que permite el acceso más rápido a la RAM, y la velocidad de la operación es un factor muy importante en el OS. Si intenta usted interrelacionar un programa en BASIC con algún tipo de código máquina, es vital que posea un conocimiento práctico de esta área de la RAM. Veamos dos aspectos importantes de la RAM del OS: los punteros del BASIC y los vectores del código máquina.



ROM a la vista
Dado que el procesador 6510 emplea direcciones de 16 bits, sólo puede direccionar un máximo de 65 536 (64 K) posiciones de memoria. El Commodore 64 tiene un total de 84 K de ROM y RAM, pero resuelve este problema de su limitación "óptica" a sólo 64 K cada vez conmutando la conexión a distintas zonas de memoria.
El "bloque" frontal representa la memoria vista normalmente por el 6510. A las áreas que quedan detrás de él se puede acceder estableciendo los registros especiales de conexión. Por ejemplo, si se escribe un programa sólo en lenguaje máquina, la ROM del BASIC probablemente no es necesaria y se puede "cegar" para que pueda ser usada la RAM de la parte trasera

De momento, supondremos que el Commodore 64 está en el modo por defecto. Lo primero que hay que remarcar es que el ordenador tiene 64 K de RAM y 20 K de ROM. De este total de 84 K, el 6510 sólo puede ver cada vez un máximo de 64 K, de aquí la necesidad de emplear la facilidad conmutación de la conexión. Por el diagrama del mapa de la memoria se puede observar que la ROM del intérprete del BASIC (el programa en código máquina que ejecuta los programas en BASIC) se encuentra entre $A000 y $CFFF, y la ROM núcleo (el código que maneja todas las funciones de entrada/salida) está entre $E000 y $FFFF.

Más abajo de estos dos bloques de ROM se hallan dos bloques de RAM de ocho K, que el 6510 no ve en modo normal, aunque una POKE (o STA) referida a estas direcciones "las atravesará" hasta llegar a la RAM subyacente. Una PEEK nos dará el contenido de la ROM superpuesta, pero la RAM puede ser leída si "desconectamos" la ROM correspondiente. Hay que insistir en que si usted desea emplear las instrucciones PEEK o POKE del BASIC, será mejor que no "desconecte" la ROM del BASIC.

Análisis del mapa de memoria del C 64

Posiciones	Empleo
$0000-$0001	Registros de control del mapa de mem. del 6510
$0002-$03FF	RAM del sistema operativo
$0400-$07F7	RAM de pantalla
$07F8-$07FF	Punteros de sprites
$0800-$9FFF	Área de progs. en BASIC usuario (incl. variables)
$A000-$BFFF	ROM del intérprete de BASIC
$C000-$CFFF	4 K de RAM libre
$D000-$D02E	Registros de control del chip VIC II (6566/9)
$D02F-$D3FF	Repetición de imágenes del VIC II
$D400-$D41C	Registros de control del chip SID (6581)
$D41D-$D7FF	Repetición de imágenes del SID
$D800-$DBE7	Cuartetos de color (i.e., cuatro bits)
$DBE8-$DBFF	Cuartetos sin usar
$DC00-$DC0F	Registros de control del chip CIA #1 E/S (6526)
$DC10-$DCFF	Repetición de imágenes de CIA #1
$DD00-$DD0F	Registros de control del chip CIA #2 E/S (6526)
$DD10-$DDFF	Repetición de imágenes de CIA #2
$E000-$FFFF	ROM del núcleo

Punteros del BASIC

Un cambio en el contenido de los punteros del BASIC puede resultar útil a veces. Por ejemplo, el contenido normal de $002B (43, en decimal) y $002C (44, en decimal) son, respectivamente, 1 y 8. Ésta es la dirección del inicio del BASIC retenida en forma *lo-hi*, y significa que el BASIC comienza en 8×256+1=2049, es decir, en $0801. El BASIC se inicia, efectivamente, en $0800, pero el OS exige que el primer byte sea siempre cero, de modo que el inicio propio de un programa en BASIC será en $0801. ($0801 es la dirección donde se inicia la operación de guardar un programa en BASIC cuando se emplea un monitor Commodore de cód. máq.)

Antes de cargar cualquier programa en BASIC es posible cambiar el límite inferior del BASIC manipulando estos dos punteros, llevando buen cuidado de que el BASIC se inicie en una frontera de página (o sea, hay que guardar el contenido de la posición $002B en uno de ellos) y que comience con un cero. (Obsérvese también que una "página" es un bloque de 256 bytes de memoria.) Así:

```
POKE2560,0:POKE44,10:NEW
```

ejecutado en el modo directo hará subir el límite inferior de la memoria dos páginas más arriba hasta $2560. Aquí NEW se usa para restablecer rápidamente todos los punteros (que ocupan las posiciones $002D a $0038). El elevar el límite inferior del BASIC puede servir para albergar en la memoria dos programas en BASIC simultáneamente. Basta con cargar el primer programa, elevar el límite inferior y después cargar el segundo.

Con más frecuencia, quizá, usted habrá de bajar la parte superior de la memoria para conseguir más espacio para el programa en leng. máq. Así,

```
POKE56,159:POKE51,0:POKE52,159
```

hará descender una página entera ese límite superior de la memoria para BASIC.

Una vez rescatado un bloque de RAM del dominio del BASIC podemos estar seguros de que el OS no empleará esta área para almacenar variables del BASIC, y por ello estará protegido nuestro código máquina (salvo de nuestros propios errores).

Tabla punteros BASIC EN CBM 64

Posiciones	Puntero
$002B-$002C	Inicio BASIC del usuario
$002D-$002E	Inicio de variables BASIC
$002F-$0030	Inicio de vectores BASIC
$0031-$0032	Fin de vectore BASIC+1
$0033-$0034	Parte inf. de las series
$0035-$0036	Utilidad
$0037-$0038	Fin BASIC del usuario

Los vectores de la RAM

El segundo bloque de RAM que reviste un interés particular para el programador es el bloque situado entre $0314 y $0333, el cual contiene los vectores de la RAM. Estos vectores son como un desvío opcional a una línea secundaria en un ferrocarril; por lo general, cuando un tren (es decir, el 6510 que ejecuta su programa, siguiendo el símil) sigue una línea pasa por el interruptor sin desviar su dirección. Pero a veces conviene desviar el tren por otra línea lateral pasando por una o dos estaciones antes de reemprender su ruta normal.

Usaremos como ejemplo el vector IRQ (solicitud de interrupción). Cada sesentavo de segundo, en un Commodore que opera normalmente, se dispara uno de los temporizadores del chip de E/S 6526 y baja la línea IRQ del 6510. Al final de su instrucción en curso, el 6510 responderá a la bajada de la línea IRQ generando una interrupción y comenzando la rutina de servicio de la IRQ, fragmento de código de mantenimiento que se inicia en $FF48. Entre otras cosas, la rutina de servicio inspecciona el teclado para detectar la pulsación de cualquier tecla. Una de las primeras cosas que hace la rutina es:

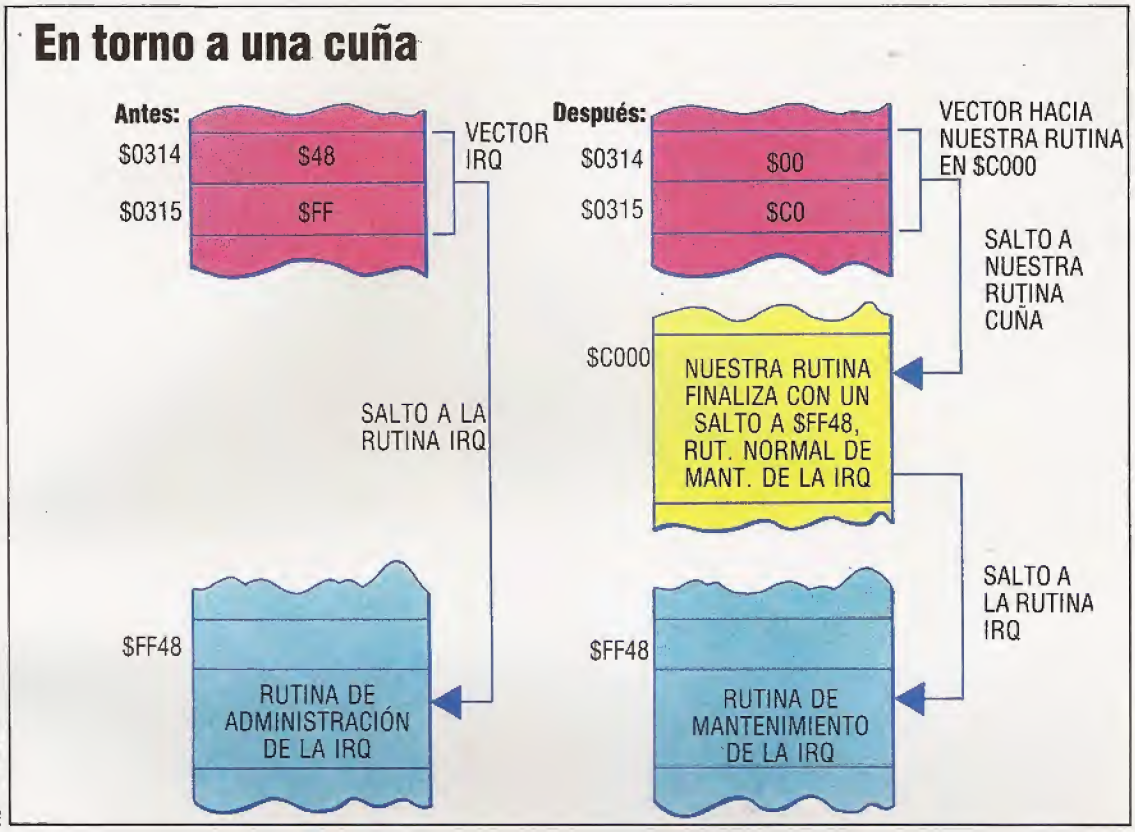
```
JMP ($0314)
```

o sea, realiza un salto indirecto a la dirección contenida en las posiciones $0314 (byte *lo*) y $0315 (byte *hi*). Estos dos bytes están en la RAM, por lo que se comienza a entrever que si cambiamos su contenido podremos apuntar a nuestra propia sección de código. Una vez ejecutado nuestro código podemos hacer que el microprocesador vuelva a apuntar a la dirección que tenía anteriormente. Así, y siempre que nuestro código no sea demasiado pretencioso (lo que excluye el uso de rutinas de núcleo) podemos conseguir que el 6510 nos realice una pequeña rutina cada sesentavo de segundo. Tal fragmento de código se denomina una *cuña*.

Tabla de vectores en el CBM 64

Posiciones	Vector
$0314-$0315	Interrupción IRQ
$0316-$0317	Interrupción BRK
$0318-$0319	Interrupción NMI
$031A-$031B	Rutina OPEN del núcleo
$031C-$031D	Rutina CLOSE del núcleo
$031E-$031F	Rutina CHKIN del núcleo
$0320-$0321	Rutina CHKOUT del núcl.
$0322-$0323	Rutina CLRCHN del núcl.
$0324-$0325	Rutina CHRIN del núcleo
$0326-$0327	Rutina CHROUT del núcl.
$0328-$0329	Rutina STOP del núcleo
$032A-$032B	Rutina GETIN del núcleo
$032C-$032D	Rutina CLALL del núcleo
$032E-$032F	Definido por el usuario
$0330-$0331	Rutina LOAD del núcleo
$0332-$0333	Rutina SAVE del núcleo

En torno a una cuña



Kevin Jones

La cuña de tiempo
Cada sesentavo de segundo es interrumpido el procesador 6510 para realizar tareas de mantenimiento tales como inspeccionar el teclado. La dirección de inicio de la rutina de mantenimiento está contenida en un par de posiciones, la $0314 y la $0315. Si se cambia la dirección contenida en estas posiciones, es posible "incrustar" nuestro propio fragmento de código máquina, que será ejecutado en lugar de la anterior rutina. Tal fragmento suele acabar con un salto a la rutina normal de mantenimiento. De este modo podemos estar seguros de que nuestro fragmento en código máquina será ejecutado cada sesentavo de segundo, al generarse la interrupción

El reloj Commodore

El siguiente listado en assembly y programa cargador de BASIC ilustra la teoría de cómo insertar una cuña o fragmento de código dentro de la rutina normal de tratamiento de interrupción IRQ. Esta cuña lleva "un reloj que da las horas" y lo visualiza en una esquina de la pantalla. El reloj, cada sesentavo de segundo, es modificado por la interrupción normal, de modo que la máquina sigue operando normalmente con el reloj en marcha, lo que permite entrar un programa en BASIC y ejecutarlo simultáneamente. La rutina establece ante todo el tiempo inicial que se le pasa, y después cambia el contenido de $0314 y $0315 para que apunten al código cuña, el cual se encarga de actualizar y visualizar el reloj cada vez que se genera una interrupción IRQ. El trabajo final consiste en realizar un salto (JMP) a una rutina normal de tratamiento de la IRQ, cuya dirección se guarda en las posiciones VECTOR y VECTOR+1.

Una alternativa para entrar y ensamblar este código fuente sería entrar el programa como una serie de sentencias DATA que constituyen el programa en código máquina. Se entra sencillamente el programa cargador de BASIC y se ejecuta para ver cómo aparece el reloj en el extremo superior derecho de la pantalla.

Se puede poner el reloj a la hora, minutos y segundos que se desea mediante POKE, que introduce la información en las posiciones 50129,50130 y 50131. Al final del programa cargador de BASIC el reloj se establece mediante las líneas 1480 y siguientes en la hora inicial 4:30:00 pm. Basta cambiar estos valores introducidos (POKE) en estas posiciones para obtener la hora que usted quiera. SYS 50138 hace que se ejecute el programa en código máquina y que empiece a funcionar el reloj. Se puede parar el reloj apretando las teclas RUN/STOP y RESTORE a la vez, o ejecutando una rutina especial que se incluye dentro del programa entrando SYS 50237

Cargador de BASIC

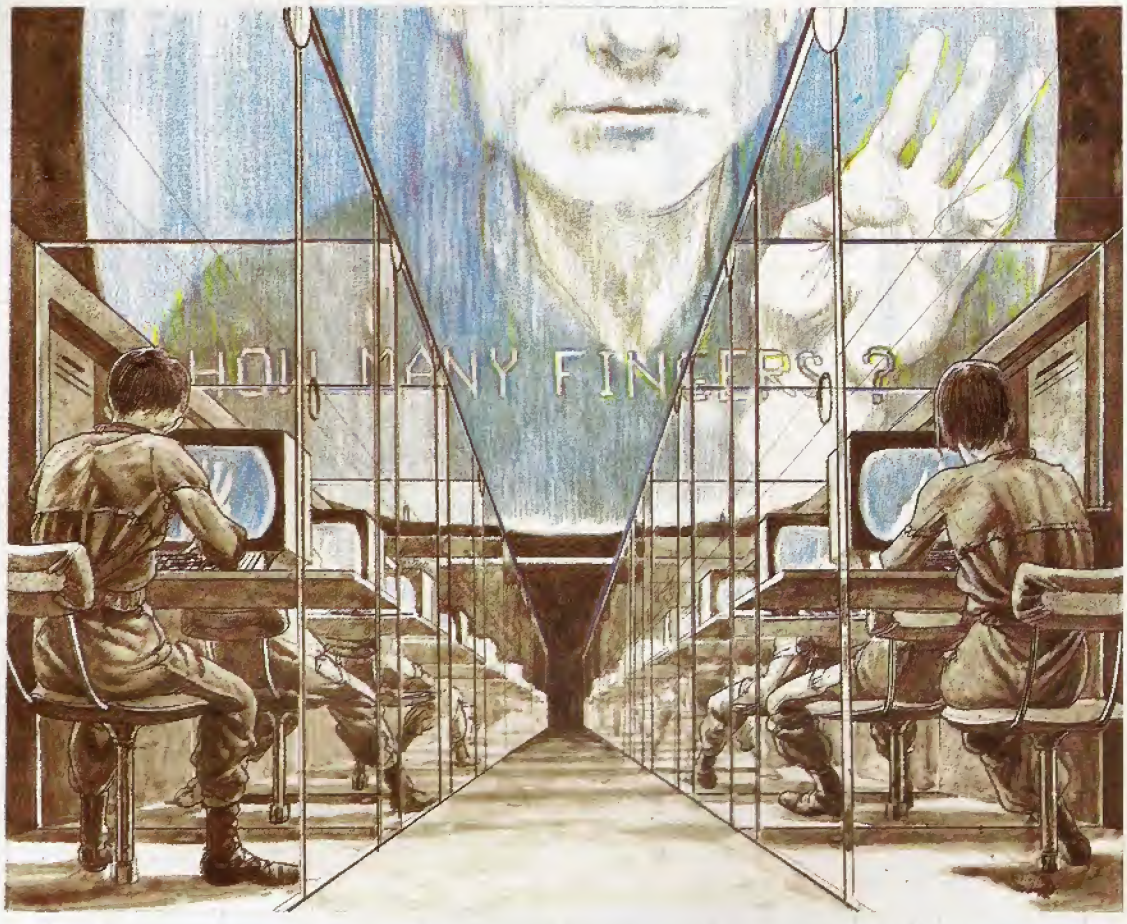
```
1000 REM ** CARGADOR EN BASIC DEL RELOJ **
1010 DATA173,166,2,240,10,169,128,13,14
1020 DATA221,141,14,221,48,8,169,127,45
1030 DATA14,221,141,14,221,169,127,45
1040 DATA15,221,141,15,221,173,208,195
1050 DATA41,128,141,208,195,173,209,195
1060 DATA32,28,197,13,208,195,141,11
1070 DATA221,173,210,195,32,28,197,141
1080 DATA10,221,173,211,195,32,28,197
1090 DATA141,9,221,169,0,141,8,221,120
1100 DATA173,20,3,141,214,195,173,21,3
1110 DATA141,215,195,169,76,141,20,3
1120 DATA169,196,141,21,3,88,96,120,173
1130 DATA214,195,141,20,3,173,215,195
1140 DATA141,21,3,88,96,173,216,195,201
1150 DATA6,240,3,76,8,197,169,255,141
1160 DATA216,195,173,213,195,240,243
1170 DATA173,11,221,170,41,128,208,5
1180 DATA169,1,76,111,196,169,16,141,38
1190 DATA4,173,212,195,141,38,216,163
1200 DATA13,141,39,4,173,212,195,141,39
1210 DATA216,138,41,16,32,14,197,141,28
1220 DATA4,173,212,195,141,28,216,138
1230 DATA32,22,197,141,29,4,173,212,195
1240 DATA141,29,216,169,58,141,30,4,173
1250 DATA212,195,141,30,216,173,10,221
1260 DATA170,32,14,197,141,31,4,173,212
1270 DATA195,141,31,216,138,32,22,197
1280 DATA141,32,4,173,212,195,141,32
1290 DATA216,169,47,141,33,4,173,212
1300 DATA195,141,33,216,173,9,221,170
1310 DATA32,14,197,141,34,4,173,212,195
```

```
1320 DATA141,34,216,138,32,22,197,141
1330 DATA35,4,173,212,195,141,35,216
1340 DATA169,46,141,36,4,173,212,195
1350 DATA141,36,216,173,8,221,105,48
1360 DATA141,37,4,173,212,195,141,37
1370 DATA216,238,216,195,108,214,195,74
1380 DATA74,74,74,24,105,48,96,41,15,24
1390 DATA105,48,96,160,255,56,200,233
1400 DATA10,176,251,105,10,141,217,195
1410 DATA152,10,10,10,10,13,217,195,96
1420 DATA42131:REM*CHECKSUM*
1430 CC=0
1440 FORI=50138TO50481
1450 READX:CC=CC+X:POKEI,X
1460 NEXT
1470 READX:IFCC<>XTHENPRINT"CHECKSUM ERROR"
1480 REM ** TEST CLOCK **
1490 POKE50128,128:REM AM/PM
1500 POKE50132,8:REM COLOR
1510 POKE50133,1:REM VISUALIZACION
1520 POKE50129,4:REM HORAS
1530 POKE50130,30:REM MINUTOS
1540 POKE50131,0:REM SEGUNDOS
1550 SYS50138:REM RUTINA DE LLAMADA

PARA DESCONECTAR LA CUÑA EMPLEAR SYS50237
EN MODO DIRECTO O DE PROGRAMA
```

Listado assembly

```
; ************************************
; *          CUÑA IRQ RELOJ          *
; *                                  *
; *                                  *
; * 50128=AM=0/PM=128                *
; * 50129=HORAS                      *
; * 50130=MINUTOS                    *
; * 50131=SEGUNDOS                   *
; * 50132=COLOR DEL RELOJ            *
; * 50133=VISUAL ON=1/OFF=0          *
; *                                  *
; * INSERCION CUÑA SYS 50138         *
; * REMOVER CUÑA SYS 50237           *
; ************************************
IRQVEC=$0314      ; VECTOR RAM IRQ
CLOCK=$DD08       ; REGISTRO TOD
D2CRA=$DD0E       ; VIA#2 CRA
D2CRB=$DD0F       ; VIA#2 CRB
PALNTS=$02A6      ; FLAG PAL/NTSC
RATE=$06          ; VISUALIZA CADA 6 IRQ
DIGIT=$30         ; CODIGO PANT. DE '0'
POINT=$2E         ; CODIGO PANT. DE '.'
SLASH=$2F         ; CODIGO PANT. DE '/'
COLON=$3A         ; CODIGO PANT. DE ':'
HZ50=$80          ; MASC. DE 50 HZ PARA TODIN
HZ60=$7F          ; MASC. DE 60 HZ PARA TODIN
AY=$01            ; CODIGO PANT. DE 'A'
PEE=$10           ; CODIGO PANT. DE 'P'
EM=$0D            ; CODIGO PANT. DE 'M'
WRITE=127         ; MASC. PARA PONER RELOJ EN CRB
SCNLOC=$041C      ; DIR. RELOJ EN PANT.
COLLOC=$D81C      ; DIR. EN MATRIZ VIDEO
TRNCLO=$0F        ; MASC. PARA CUARTETO INFERIOR

*=$C3D0           ; CODIGO DIR. INICIO
AMPM *=*+1        ; FLAG AM/PM
HOURS *=*+1       ; VALOR HORAS (INICIAL.)
MINS *=*+1        ; VALOR MINUTOS
SECS *=*+1        ; VALOR SEGUNDOS
COLOR *=*+1       ; COLOR RELOJ
DISPLY *=*+1      ; FLAG VISUAL./DESAPAR.
VECTOR *=*+2      ; ALMAC. PARA ANT. VECTOR IRQ
COUNT *=*+1       ; CONTADOR IRQ
TEMP 1 *=*+1

;
; INSERTAR CUÑA
;
   LDA  PALNTS    ; PAL O NSTC
   BEQ  NTSC      ; BIFURCACION PARA NSTC
   LDA  #HZ50     ; DEBE SER PAL
   ORA  D2CRA
   STA  D2CRA     ; ESTABLECE TODIN PARA 50 HZ
   BMI  PALDUN
NTSC
   LDA  #HZ60     ; NTSC
   AND  D2CRA
   STA  D2CRA     ; ESTABLECE TODIN PARA 60 HZ
PALDUN
   LDA  #WRITE
   AND  D2CRB     ; ESTABLECE RELOJ SIN ALARMA
   STA  D2CRB
   LDA  AMPM
   AND  #128      ; HACE UN VALOR AMPM VALIDO
   STA  AMPM
   LDA  HOURS     ; TOMA LA HORA
   JSR  BINBCD    ; CONVERSION EN BCD
   ORA  AMPM      ; OR CON FLAG AM/PM
   STA  CLOCK+3   ; ALMACENA EN RELOJ
   LDA  MINS      ; TOMA LOS MINUTOS
   JSR  BINBCO    ; CONVERSION EN BCD
   STA  CLOCK+2   ; ALMACENA EN RELOJ
   LDA  SECS      ; TOMA SEGUNDOS
   JSR  BINBCD    ; CONVERSION EN BCD
   STA  CLOCK+1   ; ALMACENA EN RELOJ
   LDA  #00       ; PONE SIEMPRE LAS DECIMAS A 0
   STA  CLOCK     ; INICIA EL RELOJ
;
   SEI            ; DESACTIVA INTERR.
   LDA  IRQVEC
   STA  VECTOR    ; GUARDA ANTIGUO VECTOR IRQ
   LDA  IRQVEC+1
   STA  VECTOR+1
;
   LDA  #<WEDGE
   STA  IRQVEC
   LDA  #>WEDGE   ; INSERTA CUÑA
   STA  IRQVEC+1
   CLI            ; ACTIVA INTERR.
   RTS
;
; QUITAR CUÑA
;
   SEI            ; DESACTIVA INTERRUPCIONES
   LDA  VECTOR
   STA  IRQVEC    ; RESTAURA VECTOR RAM
   LDA  VECTOR+1
   STA  IRQVEC+1
   CLI            ; ACTIVA INTERRUPCIONES
   RTS

;
; AQUI COMIENZA LA CUÑA
;
WEDGE
   LDA  COUNT     ;
   CMP  #RATE     ; RELOJ CON ESTA IRQ?
   BEQ  CONT      ;
OUT
   JMP  EXIT      ; NO
CONT
   LDA  #$FF      ; RESTABLECE CONTADOR IRQ
   STA  COUNT
   LDA  DISPLY    ; VISUALIZACION?
   BEQ  OUT       ; NO... BIFURCAR

   LDA  CLOCK+3   ; TOMA HORAS/AM/PM
   TAX            ; PONE UNA COPIA EN EL REG X
   AND  #$80      ; TOMA AM/PM
   BNE  PM        ; BIFURCA SI ES PM
   LDA  #AY       ; VISUALIZA 'A'
   JMP  MERIDP
PM
   LDA  #PEE      ; VISUALIZA 'P'
MERIDP
   STA  SCNLOC+10
   LDA  COLOR     ; TOMA COLOR
   STA  COLLOC+10 ; ESTABLECE EL COLOR
   LDA  #EM       ; VISUALIZA 'M'
   STA  SCNLOC+11
   LDA  COLOR     ; ESTABLECE COLOR
   STA  COLLOC+11
;
; PONE LAS HORAS
;
   TXA            ; TOMA LA HORA
   AND  #$10      ; SE DESEA EL DIGITO SUP.
   JSR  HIDIGT    ; TOMA CODIGO PANT.
   STA  SCNLOC    ; LO VISUALIZA
   LDA  COLOR
   STA  COLLOC    ; ESTABLECE COLOR
   TXA            ; TOMA EL BYTE DE NUEVO
   JSR  LODIGT    ; TOMA EL DIGITO INF.
   STA  SCNLOC+1  ; LO VISUALIZA
   LDA  COLOR
   STA  COLLOC+1  ; ESTABLECE COLOR
;
   LDA  #COLON    ; SEPARADOR HRS/MINS
   STA  SCNLOC+2
   LDA  COLOR
   STA  COLLOC+2
;
; PONE LOS MINUTOS
;
   LDA  CLOCK+2   ; TOMA MINUTOS
   TAX
   JSR  HIDIGT    ; HACE DIGITO SUP.
   STA  SCNLOC+3  ; LO VISUALIZA
   LDA  COLOR
   STA  COLLOC+3  ; Y COLOREA
   TXA            ; TOMA EL BYTE DE NUEVO
   JSR  LODIGT    ; HACE EL DIGITO INF.
   STA  SCNLOC+4  ; LO VISUALIZA
   LDA  COLOR     ;
   STA  COLLOC+4    ESTABLECE COLOR
;
   LDA  #SLASH    ; SEPARADOR MIN./SEG.
   STA  SCNLOC+5
   LDA  COLOR
   STA  COLLOC+5
;
; PONE LOS SEGUNDOS
;
   LDA  CLOCK+1   ; TOMA SEGUNDOS
   TAX
   JSR  HIDIGT    ; HACE DIGITO SUP.
   STA  SCNLOC+6  ; LO VISUALIZA
   LDA  COLOR
   STA  COLLOC+6  ; Y COLOREA
   TXA            ; TOMA EL BYTE DE NUEVO
   JSR  LODIGT    ; HACE EL DIGITO INF.
   STA  SCNLOC+7  ; LO VISUALIZA
   LDA  COLOR
   STA  COLLOC+7  ; Y COLOREA
;
   LDA  #POINT    ; SEPARADOR SEG./DECIMAS
   STA  SCNLOC+8
   LDA  COLOR
   STA  COLLOC+8
;
; PONE LAS DECIMAS
;
   LDA  CLOCK     ; TOMA EL VALOR DE DECIMAS
   ADC  #DIGIT    ; SUMA $30 PARA COD. PANT.
   STA  SCNLOC+9  ; LO VISUALIZA
   LDA  COLOR
   STA  COLLOC+9  ; Y COLOREA
;
EXIT
   INC  COUNT     ; INCREMENTA CONT. IRQ
   JMP  (VECTOR)  ; VA AL RESTO DE IRQ
;
; SUBRUTINAS
;
HIDIGT
   LSR  A
   LSR  A         ; MUEVE CUART. SUP. AL INF.
   LSR  A
   LSR  A
   CLC
   ADC  #DIGIT    ; SUMA $30 PARA COD. PANT.
   RTS

LODIGT
   AND  #TRNCLO   ; DESENMASCARA CUART. SUP.
   CLC
   ADC  #DIGIT    ; SUMA $30 PARA COD. PANT.
   RTS

;
; CONVERSION DE BINARIO A BCD
;
BINBCD
   LDY  #$FF
   SEC
D10
   INY
   SBC  #10       ; RESTA 10 HASTA -VE
   BCS  D10
   ADC  #10       ; SUMA DE 10 PARA ABAJO
   STA  TEMP 1    ; ALMACENA EL RESTO
   TYA            ; TOMA NUM PF 10S RESTADO
   ASL  A
   ASL  A
   ASL  A         ; LO DESPLAZA AL CUART. SUP.
   ASL  A
   ORA  TEMP 1    ; PONE RESTO EN CUART. INF.
   RTS
```

Impreso con el amable permiso de los autores y de Ellis Horwood LTD. Tomado de *Mastering the Commodore 64*, por Jones y Carpenter



Nick Harris

Adoctrinamiento interactivo
La imagen del CAL como precursor de una pesadilla orwelliana se debe en gran parte al trabajo de los primeros psicólogos conductistas y muy poco a los desarrollos de hoy en día. La tecnología del disco compacto, el video interactivo y las redes representan grandes promesas para el futuro del aprendizaje asistido por ordenador

¿Retorno a lo básico?

En esta ocasión nos referiremos al CAL y comentaremos las opiniones tanto de sus defensores como de sus detractores

El vocablo CAL corresponde a las siglas de *computer-assisted learning*: aprendizaje asistido por ordenador. También en algunas ocasiones se lo conoce como CAI (instrucción asistida por ordenador) o CBL (aprendizaje basado en ordenador). El término se ha utilizado en contextos notablemente diferentes para definir desde una clase determinada de software didáctico hasta cualquier actividad educativa de carácter general que implique el empleo de ordenadores. En este capítulo hablaremos del CAL en su sentido más estricto, es decir, para describir programas de enseñanza y *no* juegos, simulaciones o programación realizada por el niño. En el CAL el ordenador se utiliza para transmitir información, para constatar si ha sido comprendida, supervisar los ejercicios y las prácticas, y actuar como una máquina "inteligente" de aprendizaje.

En Gran Bretaña, la opinión actual sitúa los programas CAL en la misma línea de actividades educativas tan banales como garabatear en la pizarra o rellenar el tintero. Este bajo concepto acerca del CAL puede decirse que se generó en los primeros días del "aprendizaje programado" y el trabajo de Skinner y sus seguidores. Esta negativa reacción se ha visto reforzada por el lanzamiento, en los años subsiguientes, de programas CAL de escasa calidad, y ahora los maestros observan con gran recelo este aprendizaje. "Los programas CAL que hemos tenido en la escuela han sido pésimos", comenta Alan Coode, quien obtuviera el premio del periódico británico *Sunday Times* a la promoción del alfabetismo informático en la educación. "Se basan en una superada filosofía de la educación y consideran el niño en términos conductistas. Se necesitaría esgrimir algún argumento muy efectivo para que me decidiera a comprar otro programa CAL."

La historia del desarrollo del CAL no es alentadora. A comienzos de 1980, con un parque de menos de 100 microordenadores en las escuelas británicas, era evidente que no había mercado para el software educativo. Los maestros que tenían acceso a los ordenadores debían escribir ellos mismos sus programas en BASIC y, dado que no eran programadores experimentados, su software era, por adjetivarlo de alguna forma, primitivo. Un maestro se pasó días escribiendo un programa que generaba

operaciones de suma, resta, multiplicación y división. Si el niño calculaba la respuesta correcta, en la pantalla aparecía un enorme tilde; si el resultado era incorrecto, se visualizaba una cruz. Los niños que normalmente odiaban "hacer sumas" disfrutaban haciéndolas en el ordenador. El programa obtuvo un enorme éxito e, independientemente de cuán poco imaginativos, amables y plagados de errores fueran los programas, a los niños les resultaba agradable utilizarlos. Lo que los cautivaba era el medio, no el mensaje. Pero a nadie se le ocurrió pensar que los niños estaban empleando un recurso muy caro para una actividad que se podía realizar igualmente con lápiz y papel.

Movidos por el interés de mantenerse a tono con los tiempos, muchos maestros han aceptado programas CAL que, en otras circunstancias, probablemente condenarían. A consecuencia de este entusiasmo mal dirigido y de la manipulación comercial, a comienzos de los años ochenta se produjo un alud de software CAL. En 1982, los programas de enseñanza, incluyendo ejercicios de destreza y práctica, representaban casi el 80 % del *School micro directory*, uno de los principales catálogos de software educativo de Estados Unidos. Esta tendencia experimentó enseguida un retroceso. La oposición que suscitó el CAL la resumió David Chandler en su libro *Young learners and the microcomputer* (Los jóvenes estudiantes y el microordenador): "El microordenador —señala— es una herramienta de asombroso poder que está haciendo posible que la práctica educativa dé un gigantesco paso atrás, al s. XIX. Es la última arma de quienes desean 'retornar a lo básico', permitiéndoles procesar niños de 'entrada' a 'salida' en función de 'objetivos behavioristas' y 'control de calidad...'"

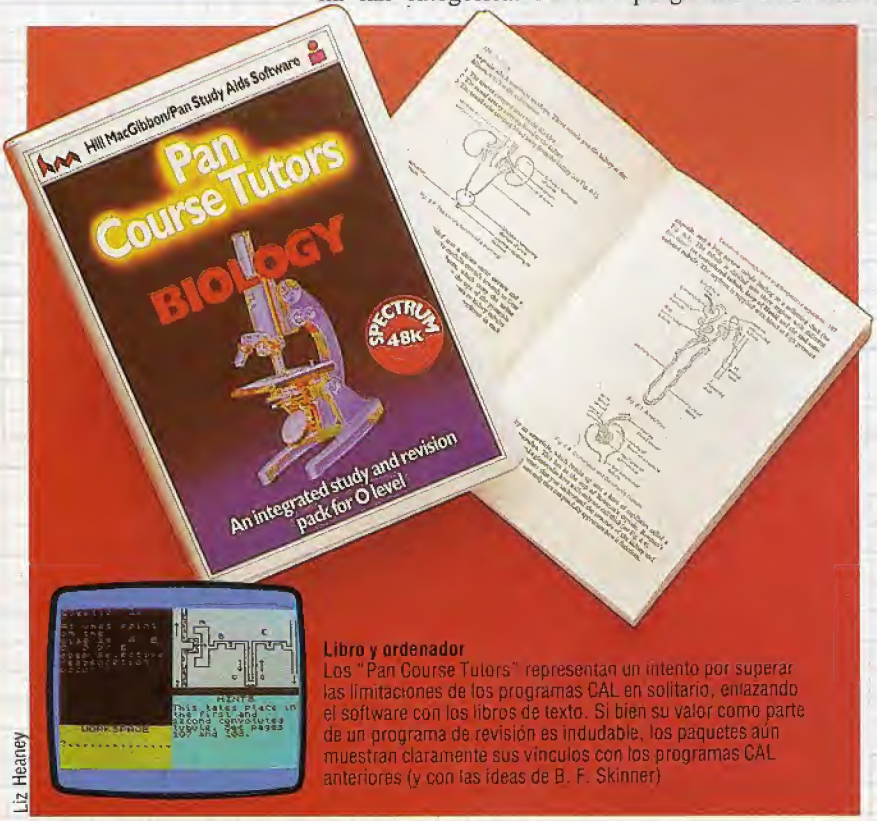
Lamentablemente, gran parte del material CAL ofrece enormes evidencias que apoyan esta condena tan categórica. Muchos programas refuerzan malos hábitos, favorecen respuestas no imaginativas y no dan más opciones que ofrecer una respuesta correcta o incorrecta. Aunque se afirma que el CAL permite que el niño aprenda a su propio ritmo, sería más exacto decir que el niño aprende al ritmo del ordenador.

No obstante, hay signos de un resurgimiento del interés por el aprendizaje asistido por ordenador. La mayor parte de los ordenadores personales los emplean varones de entre 11 y 18 años para practicar juegos de tipo recreativo, y este mercado ha ayudado a establecer una gran base de software que se utiliza ampliamente en el campo de la educación. Como extensión, recientemente ha aparecido un enorme mercado para software CAL, como apoyo al estudio en casa para preparar los exámenes escolares. Una de las principales firmas británicas de software educativo, Hill MacGibbon, ha sabido reconocer este mercado nuevo y ha invertido en la producción de algunos estimables programas CAL de apoyo a los planes de estudio para la preparación de exámenes.

Al igual que otras empresas de software, Hill MacGibbon ha identificado dos áreas principales del software educativo. La primera es la del paquete diseñado para un fin específico, al objeto de preparar al alumno para exámenes académicos o profesionales. La segunda área está relacionada con la educación en su sentido más amplio, en la cual los estudiantes pueden utilizar el ordenador para explorar temas de otro modo inaccesibles.

Un buen ejemplo de la primera categoría de productos es la serie "Pan Course Tutors", editada conjuntamente por Pan Books y Hill MacGibbon. La serie se compone de seis títulos (*Matemáticas, Biología, Química, Economía, Francés* y *Física*) y cada uno viene acompañado por un libro de texto. Ello evita el problema de intentar incluir en el programa grandes cantidades de texto, dejando al software libre para concentrarse en el aspecto interactivo del aprendizaje. El formato de libro/programa muy probablemente persistirá mientras los usuarios de micros personales dispongan de máquinas de ocho bits; pero a medida que se vayan introduciendo en los hogares procesadores capaces de direccionar más memoria, los libros de texto se irán volviendo, en gran medida, innecesarios. La perspectiva más probable es que el software con ventanas permita la presentación de texto y la interacción con el alumno en la misma pantalla.

En la serie "Pan Course Tutors", los programas formulan preguntas sobre la materia elegida. Entre los beneficios del uso del ordenador se incluye la ventaja psicológica de no poder recurrir a la página donde vienen las respuestas y, si se da una respuesta incorrecta, se guía al alumno para hallar la respuesta correcta, con una pista gráfica o un mensaje como "¿Te has acordado de incluir los paréntesis?". Si se adelanta una segunda respuesta equivocada, se visualiza otra pista, por lo general una referencia a una página o sección del libro de texto, para que el estudiante pueda consultar las referencias. Las respuestas se cronometran y se registran, de modo que un programa puede analizar los resultados e incluso sugerir áreas susceptibles de un estudio más profundo. Al trabajar alternativamente con el ordenador y el libro de texto, los estudiantes pueden aprender a su propio ritmo y tomándose el tiempo que necesiten, sin que haya competitividad.



Libro y ordenador
Los "Pan Course Tutors" representan un intento por superar las limitaciones de los programas CAL en solitario, enlazando el software con los libros de texto. Si bien su valor como parte de un programa de revisión es indudable, los paquetes aún muestran claramente sus vínculos con los programas CAL anteriores (y con las ideas de B. F. Skinner)

Liz Heaney

El aprendizaje asistido por ordenador también se puede utilizar para enseñar a los niños a usar el micro como una herramienta. Un programa que guíe al niño a través de las diversas funciones de un procesador de textos, por ejemplo, tiene un enorme valor. Es más fácil seguir las indicaciones en la pantalla, digitando las instrucciones adecuadas y obteniendo realimentación, que ir recorriendo laboriosamente el manual. El Apple Macintosh emplea este tipo de presentación en disco, en unión con una cassette de audio, para enseñar las características del ordenador, el procesador de textos y los programas para gráficos.

Lamentablemente, el crecimiento del mercado potencial de software educativo ya ha suscitado algunas extravagantes consideraciones sobre la pobreza de los productos, lo que, como cabe suponer, inhibe el desarrollo de software de calidad. Una conocida editorial creó un costoso paquete compuesto por cuatro programas, uno de los cuales era un juego de *rounders* (juego de pelota parecido al béisbol). Según la nota que lo acompañaba, éste tenía por objetivo su utilización durante un día de lluvia para enseñar a los niños las reglas del juego. Pero debido a que el programador había tomado tantos atajos al diseñar el paquete (incluyendo la omisión de muchos de los puntos más delicados de las reglas), probablemente los niños que intenten aprender a jugar al *rounders* con este paquete lo tendrán muy difícil. Entre otras afirmaciones excesivas que se hicieron sobre sus beneficios educativos se incluía la siguiente: "El programa ofrece muchísimas oportunidades para la digitación en el teclado... Comienza al pedir a uno de los dos jugadores que entre su nombre." También afirma estimular el interés por los deportes. El programa de juego está escrito en BASIC y es tan lento que incluso carece de la emoción de un videojuego sin valor educativo.

Cuando se agote la novedad que suponen los ordenadores, los maestros y los especialistas en educación considerarán más críticamente el software y expresarán sus necesidades con mayor claridad. Probablemente en el futuro veamos menos programas de aprendizaje asistido por ordenador pero de un estándar más elevado, útiles en áreas específicas de los planes de estudio o para personas que estudien en casa.

La línea divisoria entre lo que se define como aprendizaje asistido por ordenador y el resto de software educativo (con el cual el niño tiene más control sobre lo que está sucediendo) no existe en la realidad. La presentación didáctica se puede combinar con simulaciones y experimentos que, de otra forma, sería imposible llevar a cabo en el laboratorio de la escuela.

Simular la realidad

El software CAL actual es útil para enseñar lo que miden los "tests" estandarizados, pero este software aporta muy poco para que el educando ponga en juego su imaginación. Para que el CAL del futuro sea un medio auxiliar atractivo y valioso para la educación, es imprescindible que cambien tres factores. Es preciso descartar las técnicas educativas anticuadas, como el aprendizaje en base a respuestas correctas, en el que se fundamentan tantos programas CAL. Se necesitan ordenadores más potentes; los microordenadores actuales, con su limitada memoria, no pueden soportar la gama de software o potencia necesaria para el aprendizaje interactivo asistido por ordenador. Pero el factor más restrictivo ha sido que el diseño de los programas ha estado en manos de quienes comprenden la programación, que no son necesariamente quienes mejor saben cómo aprende la gente.

Un primer intento por cambiar esta situación fue el PILOT, un lenguaje para ordenador diseñado para permitir a los maestros la creación de paquetes de aprendizaje para niños. En el Centro de Investigación de Xerox de Palo Alto, California, se han continuado los trabajos en esta área. Allí se ha desarrollado un Rehearsal World (mundo de ensayo) utilizando un entorno de programación visual. Fueron los trabajos llevados a cabo en este laboratorio los que llevaron al desarrollo del Apple Macintosh, y Rehearsal World parece ampliar el concepto que subyace tras la máquina. Quienes no sean programadores pueden utilizar Rehearsal World para producir software educativo en SMALLTALK. El diseñador mueve "actores" a través de un "escenario", enseñándoles cómo interactuar enviándoles "pies", todo ello en un entorno gráfico interactivo. Se intenta utilizar este entorno de un modo similar a los gráficos de tortuga del LOGO. Rehearsal World necesita muchísima memoria, pero marca una de las direcciones que seguirá el CAL en el futuro.

Incluso en la actualidad, un ordenador ofrece al alumno muchas rutas a través de un programa de aprendizaje, y las posibilidades potenciales del empleo del video interactivo son enormes. La combinación de un lenguaje creativo como el MICROTEX o el Apple SUPER PILOT con el video interactivo proporcionaría al educador un control sin precedentes sobre la situación de aprendizaje.

En la escuela, el video interactivo se puede utilizar para impartir clases, proporcionando una fuente de sonido e imagen, o bien lo pueden usar grupos reducidos de niños que empleen el disco como preceptor. El aprendizaje individualizado empleando todas las capacidades interactivas para el estudio abierto es, ciertamente, una posibilidad

Tony Stone

En plena actuación
La difusión de nuevos sistemas operativos de "entorno gráfico", como el Macintosh de Apple (abajo) y el GEM de Digital Research está poniendo un considerable poder de proceso bajo el control de usuarios que anteriormente eran analfabetos desde el punto de vista informático. Las implicaciones de las nuevas máquinas "amables con el usuario" respecto al aprendizaje asistido por ordenador son enormes, porque anteriormente todos los programas CAL debían trabajar sobre el supuesto de que el usuario no tenía ningún control sobre la máquina como no fuera responder a una pregunta cuando así se le pidiera

Hombre al agua

El Módulo 6 de nuestro juego de simulación se ocupa de algunos de los sucesos y vicisitudes a los que puede verse enfrentada la tripulación durante la larga travesía del océano

Para poder simular los sucesos y vicisitudes que ocurrían durante los largos viajes marítimos del s. XVI, debemos preparar el programa para que produzca un cierto número de eventos. Algunos de estos eventos pueden ser de ayuda para los marineros, como que sople un viento favorable, pero otros pueden tener efectos nefastos para la travesía, como sería que algunos miembros de la tripulación o algunas provisiones se cayeran al agua.

Hay un total de 16 eventos posibles, sólo dos de los cuales se pueden producir en una semana dada. En este capítulo escribiremos una subrutina para seleccionar al azar un suceso de la lista de contingencias, incluyendo el código para las cinco primeras contingencias. Anteriormente habíamos insertado un trozo de código para establecer al azar las tasas de fortaleza de la tripulación. Antes de digitar el Módulo 6, suprima las líneas de la 601 a la 604. El nuevo código se compone de varias partes: una sección de inicialización, dos líneas en el bucle principal del viaje y una subrutina para seleccionar un evento al azar y manejar cinco de éstos.

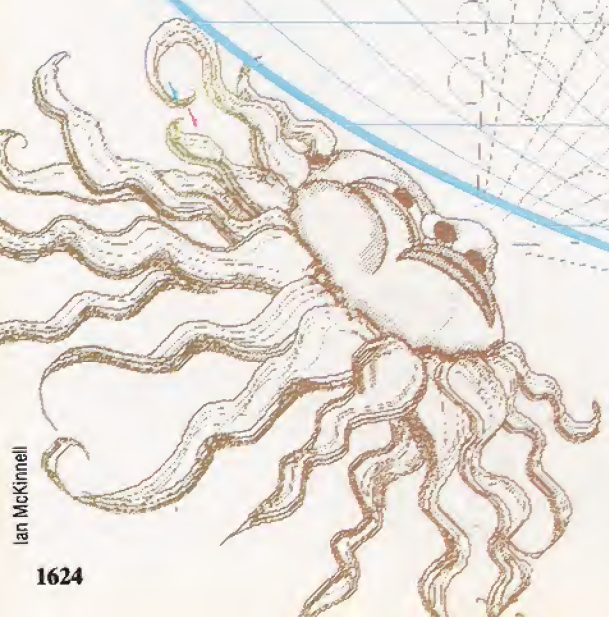
Es importante que cada evento aleatorio se produzca una sola vez durante el viaje. Para conseguirlo, en la línea 42 se DIMensiona una matriz, RR(), que contiene 16 elementos que corresponden a los 16 eventos posibles. Cada elemento se establece en 0. Cuando se produzca un evento, el elemento correspondiente se establecerá en 1, de modo que el programa lo reconocerá y no lo repetirá. Las variables que indican coordenadas dentro de un programa se denominan *flags* o *banderas*. Es necesario llevar la cuenta de los eventos a medida que se vayan produciendo para evitar que el programa busque incesantemente otro evento en la matriz, una vez que todos ellos se hayan producido. La cantidad de eventos se registra en la variable RC, que se incrementa cada vez que se genera uno. El número de posibles eventos está retenido en RM. Observe que, como en el próximo capítulo iremos añadiendo más eventos, en el futuro será necesario cambiar el valor de RM para hacerlo corresponder a la nueva cantidad.

En el bucle principal del viaje, la línea 860 llama a la subrutina de la línea 5500, que genera un evento al azar. Esta llamada está insertada en el bucle principal de modo que el evento se genere después de que el Diario de Navegación del Capitán haya dado la duración estimada del viaje. La subrutina 5500 selecciona cuál de los eventos se producirá cada semana. Si ya han acontecido todos, el juego retorna al programa principal sin generar ningún evento. La línea 5502 comprueba esto mediante la comparación de RC, el número de eventos que ya se han producido, con RM, el número de eventos del programa.

Eventos aleatorios

En la línea 5510 se genera un número al azar entre 1 y el número de eventos posibles, RM. La línea 5520 comprueba en la matriz RR() si el elemento para ese evento se ha establecido en 1, lo que significa que ya se ha producido. De ser así, retorna al programa principal sin generar ningún evento. Cuando se ha seleccionado un evento nuevo, la línea 5523 establece en 1 la bandera correspondiente en la matriz RR(), para impedir que el mismo vuelva a ser seleccionado. La misma línea incrementa en uno el contador de acontecimientos, RC.

En la línea 5525 se utiliza la sentencia ON X GOTO para dirigir el programa al código apropiado para el evento seleccionado. ON...GOTO y ON...GOSUB son,



Ian McKinnell

juntas, una forma de reemplazar un número de sentencias IF...THEN. Cada instrucción va seguida por una lista de números de línea. Si la variable es 1, se selecciona el primer número de línea de la lista. Si su valor es 2, entonces se selecciona el segundo número de línea, y así sucesivamente. Observe que el Spectrum carece de ON...GOTO y ON...GOSUB. Para este módulo habrá de remitirse al recuadro *Complementos al* BASIC.

Tenemos cinco sentencias correspondientes a los cinco eventos que estamos cubriendo. El primer evento, una persona que es arrastrada por la borda, se produce en la línea 5540. Los últimos cuatro números de la lista ON...GOTO son los mismos, de modo que tanto si el número aleatorio es 2, 3, 4 o 5, se envía el programa al mismo sitio. Las cuatro contingencias de los elementos del 2 al 5 consisten en provisiones que caen por la borda, y las cubre el mismo trozo de programa que comienza en la línea 5570.

Si se selecciona el primer número de línea, se imprime un mensaje que informa que una persona ha caído al agua. La línea 5554 imprime el nuevo total de la tripulación, restando 1 a CN, la variable que representa el número de tripulantes, pero no altera el valor de CN en esta etapa. Ello se hace durante el informe de final de semana, comprobando si la tasa de fortaleza de algún miembro se ha establecido en −999.

Por supuesto, debe elegirse qué tripulante caerá al agua. La selección se efectúa en el bucle de la línea 5558, que busca en la matriz de categoría/fortaleza de la tripulación, desde el principio, ignorando todos los elementos establecidos en 0 o −999, y seleccionando como víctima al primer tripulante vivo. La línea 5560 arroja éste al mar, estableciendo su tasa de fortaleza en −999. El valor de T se establece entonces en 16, para hacer que el programa salga del bucle.

Los restantes eventos de los que nos ocupamos en este módulo, etiquetados del 2 al 5 en la matriz RR(), tratan de los cuatro tipos de provisiones que caen al agua. Para hacer más interesante el juego, se selecciona una cantidad aleatoria de cada provisión para lanzar por la borda. La cantidad de fruta, verduras, carne y agua se representa mediante PA(1), PA(2), PA(3), PA(4).

En la línea 5572 se le resta 1 al número aleatorio, X, generado por la subrutina 5500, ya que al perder un tripulante ya se ha cubierto un evento. El número aleatorio estará ahora entre 1 y 4, correspondiendo a uno de los cuatro tipos de provisiones. La línea 5574 comprueba si la provisión está agotada y, si así es, no emprende ninguna acción. El programa informa, entonces, al jugador que alguna de las provisiones se ha ido por la borda.

La cantidad de comida o agua perdida puede ser una tercera, cuarta o quinta parte de las existencias. La fracción se elige al azar en la línea 5592, dividiendo la cantidad de provisiones restantes, PA(), por tres, cuatro o cinco, y restando esa fracción a la cantidad original. El programa calcula luego la cantidad de raciones semanales que quedan de esa provisión restante, mediante la ecuación de la línea 5594.

Se puede ejecutar el programa con las cinco contingencias ya mencionadas. Todas ellas tienen un efecto negativo sobre el viaje, poniendo más dificultades para llevarlo a cabo con éxito.

Módulo 6: Eventos al azar

Inicializar variables eventos

```
42 DIM RR(16)
43 REM INDICADORES PARA MOSTRAR SI YA HA OCURRIDO EL
   EVENTO AZAR(N)
44 RC=0
45 REM CONTADOR EVENTO AL AZAR HASTA AHORA
46 RM=5
47 REM NUM. DE EVENTOS ALEATORIOS DEL PROGRAMA
```

Adición al bucle principal del viaje

```
860 GOSUB 5500
861 REM GO TO GENERAR EVENTO AL AZAR
```

Subrutina de eventos aleatorios

```
5500 REM GENERADOR DE EVENTOS AL AZAR
5502 IF RC=RM THEN RETURN
5503 REM SALIR SI TODOS LOS EVENTOS YA PRODUCIDOS
5504 PRINTCHR$(147)
5505 GOSUB 9200
5510 X=INT(RND(1)*RM)+1
5515 REM GENERAR NUM. AL AZAR ENTRE 1 y RM
5520 IF RR(X)=1 THEN RETURN
5222 REM RETURN SI ESTE EVENTO YA SE PRODUJO
5523 RR(X)=1:RC=RC+1
5524 REM ESTABLECER INDIC. PARA SEÑALAR QUE ESTE EVENTO
     YA SE PRODUJO E INCREMENTAR CONTADOR EVENTOS
5525 ON X GOTO 5540,5570,5570,5570,5570
5528 REM IR A CODIGO APROPIADO PARA ESTE EVENTO
5530 PRINT:S$=K$:GOSUB 9100
5535 GET I$:IF I$=""THEN 5535
5539 RETURN:REM RETORNAR A BUCLE PRINCIPAL VIAJE
5540 REM EVENTO 1 – HOMBRE AL AGUA!
5542 PRINT
5543 S$="  DURANTE LA SEMANA*":GOSUB 9100
5545 PRINT:GOSUB 9200
5546 S$="   1 PERSONA CAYO POR LA BORDA*":GOSUB 9100
5548 S$="  DURANTE UNA TORMENTA*":GOSUB 9100
5550 PRINT:GOSUB 9200
5552 S$="TU TRIPULACION SE HA REDUCIDO AHORA A*":
     GOSUB 9100
5554 PRINT CN−1;"MIEMBROS"
5558 FOR T=1 TO 16
5559 REM BUSCAR QUE TRIPULANTE SE PERDIO
5560 IF TS(T,2)=0 OR TS(T,2)=−999 THEN 5566
5562 TS(T,2)=−999:REM MUERTO
5564 T=16
5566 NEXT
5568 GOTO 5530
5570 REM EVENTOS DEL 2 AL 5 – PERDIDA DE PROVISIONES
5572 X=X−1:REM AHORA X INDICA LA PROVISION (1−4)
5574 IF PA(X)=0 OR PA(X)=−999 THEN 5530
5576 REM NINGUNA ACCION SI YA SE HA AGOTADO ESTA
     PROVISION
5578 PRINT
5580 S$="  DURANTE LA SEMANA*":GOSUB 9100
5582 PRINT:GOSUB 9200
5584 PRINT"  PARTE DE TU ";P$(X)
5586 S$="  FUE ARRASTRADA POR LA BORDA*":GOSUB 9100
5588 PRINT:GOSUB 9200
5590 S$="AHORA TIENES APROXIMADAMENTE*":GOSUB 9100
5592 PA(X)=PA(X)−INT(PA(X)/(INT(RND(1)*3)+3))
5593 REM REDUCIR CANTIDAD PROV EN 1/2 1/3 o 1/4
5594 PRINT INT(PA(X)/(CN*PN(X)));
5595 PRINT "Y TE QUEDA PARA ";P$(X);"  SEMANAS
     APROXIMADAMENTE"
5599 GOTO 5530
```

Complementos al BASIC

Spectrum:
Efectuar las siguientes modificaciones:

```
5504 CLS
5525 IF X=1 THEN GOTO 5540
5526 IF X>1 AND X<6 THEN GOTO 5570
5535 LET I$=INKEY$:IF I$=""THEN GOTO
     5535
```

BBC Micro:
Efectuar las siguientes modificaciones:

```
5504 CLS
5535 I$=GET$
```

Capturando "klingons" en 17 dimensiones

Hablaremos del uso de las matrices, incluyendo aspectos tales como empaquetamiento, índices, etc.

En muchos lenguajes de programación la matriz constituye un medio muy natural de asociar datos del mundo real tales como listas, tablas y matrices a una estructura de datos para ordenador, y una forma eficaz de facilitar el acceso a elementos individuales para su proceso. No obstante, la principal causa de su uso casi universal es producto sólo del hábito. Sencillamente, la mayor parte de los lenguajes primitivos no disponían de ningún método de estructuración de datos aparte de la matriz. El primero de los lenguajes de alto nivel, el FORTRAN, sólo tenía números complejos y matrices, y su descendiente, el BASIC, omitió los números complejos. Éste es el motivo por el cual el bucle FOR...NEXT es la única estructura de iteración definida.

Todo ello estaba muy bien en los días en que el FORTRAN se empleaba sólo para efectuar operaciones de matrices sobre todos los elementos de una matriz, y cuando el BASIC se utilizaba sólo para introducir a los estudiantes en los rigores de la programación en FORTRAN. Ya hemos visto algunos ejemplos de la flexibilidad extra de control que le confieren al PASCAL los bucles activados por condición (WHILE y REPEAT). Y, lo que quizá sea aún más importante, estamos comenzando a vislumbrar parte del inmenso potencial y sencillez de expresión que ofrecen las otras estructuras de datos del PASCAL, como los conjuntos y los registros. En PASCAL, por consiguiente, el dominio de la matriz como una "herramienta de datos" universal se ve considerablemente reducido.

Matrices flexibles

Si bien la mayoría de los programadores suelen sentirse en terreno conocido cuando se enfrentan con las matrices del PASCAL, hay dos puntos significativos que se han de tener siempre presentes. Debido a que el PASCAL posibilita una elección tan amplia de los métodos para estructuración de datos, muchos problemas de programación se pueden muy bien resolver con más eficacia mediante el empleo de otras estructuras. En segundo lugar, las matrices no están limitadas por ninguna restricción en cuanto a la complejidad estructural de sus elementos, de modo que en PASCAL se puede obtener una flexibilidad mucho mayor.

La definición de un tipo de matriz reserva espacio de almacenamiento para un cierto tamaño fijo de matriz, y define tanto el tipo base de su índice como el tipo de cada componente de la misma. De modo que, por ejemplo:

```
TYPE
  ContCaract=ARRAY['A'..'Z'] OF integer;
VAR
  lista : ContCaract;
```

reservaría almacenamiento para 26 enteros que serían referenciados especificando el identificador de matriz seguido por una expresión de indexación entre paréntesis. Con las matrices siempre se utilizan corchetes, tal como se hacía originalmente en BASIC (el posterior empleo de los paréntesis para el subíndice se debió sólo al hecho de que algunos juegos de caracteres limitados carecían de corchetes). Para acceder a un determinado entero de la lista que acabamos de mencionar, podemos escribir:

```
lista ['M'] o lista [pred(simbolo)]
```

En el segundo ejemplo, la expresión pred (simbolo), por supuesto, debe tener un valor char comprendido entre el subrango de la 'A' a la 'Z'. Para indexar una dimensión de matriz se puede emplear cualquier tipo escalar, pero no tipos estructurados ni reales. Esto impide tentativas absurdas de referirse a lista ['segundo'] o bandera[3.74], por ejemplo. Utilizando la definición de tipo ilustrada, podemos inicializar a cero todos los elementos del contador de una matriz de tipo ContCaract mediante:

```
VAR
    letra      :char;
    contador   :ContCaract;
BEGIN
  FOR letra :='A' TO 'Z' DO
    contador [letra]:=0
```

y así sucesivamente. Obviamente, contador[1] es ilegal para esta estructura (es un tipo de índice incorrecto) y contador ['a'] no existe (el subíndice está fuera del subrango definido), de manera que es una buena costumbre definir identificadores de tipo para las variables de indexación. En cualquier caso, nos encontraremos invariablemente con que los necesitamos cuando definimos nuestros propios procedimientos y funciones.

Si, a su vez, estos subrangos están definidos con valores dados en una definición CONST, sería posible reescribir todo el programa para tratar con estructuras de distinto tamaño, mediante la alteración de una sola línea. Por ejemplo, si bien en PASCAL no existe ningún tipo serie predeclarado, una forma de producirlo (aunque no necesariamente la mejor) sería:

```
CONST
  LongSerie=25;
TYPE
  TamañoSerie=1..LongSerie;
  serie=PACKED ARRAY [TamañoSerie] OF char;
```

¡Típico!
La disciplina estructuración de datos del PASCAL obliga al programador a adoptar un enfoque más metódico en la construcción del programa. Sólo dos de los tres tipos de datos básicos, como vemos en el diagrama, se pueden subdividir en otras categorías. Pero la gama de posibles datos estructurados y simples exige definiciones estrictas de todos los datos que se utilizan en un programa. Con frecuencia, este esfuerzo adicional ofrece al programador de PASCAL la recompensa de soluciones más elegantes a los problemas de la programación que las que pueden proporcionar otros lenguajes

Observe que la palabra reservada PACKED puede preceder a una palabra reservada de cualquier tipo estructurado (SET, ARRAY, RECORD o FILE). Al "empaquetar" estructuras de datos, podemos obtener importantes beneficios en el empleo de memoria, especialmente en ordenadores con grandes tamaños de palabra (16/32 bits o más).

El tipo "serie"

El único momento en el que uno necesita verdaderamente ser consciente del empaquetamiento es cuando se emplean constantes literales en serie. Una serie de caracteres encerrada entre comillas se supone indexada a partir del uno (no del cero) y, por consiguiente, queda declarada implícitamente como:

```
PACKED ARRAY[1..N] OF char
```

donde N es el número de caracteres de la serie. Analicemos este programa:

```
PROGRAM SerieEmpaq;
CONST
  Pascal='Pascal';
TYPE
  serie=PACKED ARRAY[1..10] OF char;
VAR
  S       :serie;
BEGIN
  S:=Pascal;
  S:='Demasiados caracteres';
  S:='Pascal  '
END.
```

En el cuerpo del programa, las dos primeras sentencias son ilegales debido a la incompatibilidad de longitudes, pero la tercera es aceptable (utilizándose cuatro espacios como caracteres de relleno). Si la definición del tipo serie no estuviera empaquetada, todas las asignaciones de literales serían incompatibles. En la práctica, estas restricciones no constituyen un problema serio. Aunque oficialmente el PASCAL sólo soporta el uso de los procedimientos read y write para E/S de estructuras completas desde y hacia archivos del almacenamiento externo, podemos escribir tipos en serie de variables o literales como éstos directamente en la VDU. Intente escri-

bir un programa utilizando una declaración de serie similar a la del último ejemplo, e incorpore la sentencia:

```
REPEAT
  write ('Serie(Q para salir):');
  ReadLn (S);
  WriteLn ('La serie era:"',s,'"')
UNTIL S[1] IN ('Q','q')
```

Una vez que logre ponerlo en funcionamiento, pruebe el programa e investigue los siguientes puntos:

- ¿Se ignoran los espacios delanteros o tabulaciones?
- ¿Qué ocurre cuando la línea es más larga que la longitud de la serie?
- ¿Qué contiene s cuando se entra un solo carácter?
- ¿Qué ocurre si sólo se pulsa Return?
- ¿Puede usted incluir un código adecuado para rellenar con espacios los elementos sin importancia?

Por cuanto concierne al PASCAL, no existe ningún límite en absoluto respecto al tamaño ni al número de dimensiones que uno puede especificar para una matriz. Si usted puede imaginar cómo se capturarían los *klingons* en un medio continuo de espacio-tiempo en 17 dimensiones, ¡entonces puede crear una estructura de datos adecuada en PASCAL! Sin embargo, habrá de disculpar a su ordenador si no recibe con agrado estas definiciones de tipos:

```
GranTipo=ARRAY [integer] OF SET OF char;
  {una demanda de al menos 64 K×128 bytes!}
AunMayor=ARRAY[1..1000] OF
    RECORD
      apellido,
      nombre : serie;
      direccion : ARRAY [1..5] OF serie;
      presente : SET OF asistencias;
      {etc.}
  END;{AunMayor}
```

El tamaño físico de la memoria limitará de modo inevitable una programación tan ambiciosa. Nuevamente ello refuerza el valor de poder especificar subrangos de los tipos escalares, minimizando así toda sobrecarga de almacenamiento.

La criba de Eratóstenes

Este programa muy conocido para hallar números primos se optimiza, desde el punto de vista de velocidad de ejecución, ignorando los números pares y utilizando una matriz de "banderas" (8 192 elementos para primos hasta 16 384). Por tanto, se requieren al menos 8 K de memoria, quizá 32 K en lenguajes que carezcan de variables booleanas de un solo byte y enteras de cuatro bytes. Por este motivo no se ejecutará sin modificar en BASIC BBC. En PASCAL, sin embargo, el conjunto completo requeriría sólo 16 384 bits (2 K) y nos permite reproducir el algoritmo original de Eratóstenes con mayor claridad y precisión:

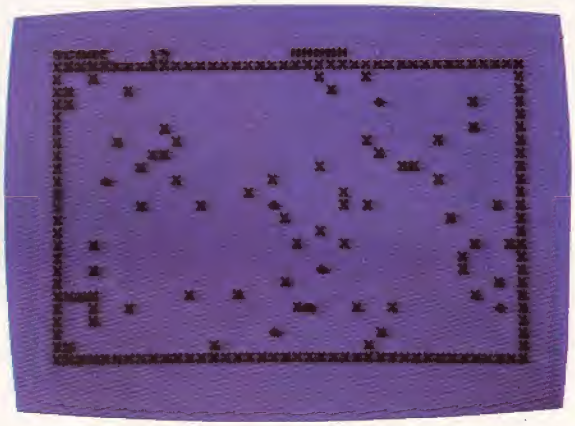
```
Colocar todos los números (1...max) en una
  criba
Sacar el número uno (primero por definición)
{imprimir si fuera necesario}

REPEAT
  Sacar el número más bajo de la criba
  {imprimir este primo}
  extraer todos los múltiplos de la criba
UNTIL la criba quede vacía
```

Sólo en potentes ordenadores centrales se suele disponer de un conjunto tan grande, pero podemos simular uno mediante el empleo de una matriz de conjuntos más pequeños (100 o 1 000 elementos, supongamos, según su compilador). El índice de la matriz de cada conjunto se halla a partir de la división entera del número, y su pertenencia al conjunto se representa mediante N módulo 100 o 1 000. Sin embargo, a menos que posea un ordenador de 100 bits, probablemente la velocidad será inferior que en la versión de matriz

```
PROGRAM     ListaCriba  (output);
{Hallar numeros primos por algoritmo de Eratostenes}
CONST
  Tamaño Conj       =    100; {~implementacion}
  PredTamañoConj    =     99; {TamañoConj-1}
  Primo Max         = 16383; {para MaxInts=32767}
  MaxLista          =    163; {PrimoMax DIV TamañoConj}

TYPE
  GamaPrimo         = 1..PrimoMax;
  dimension         = 0..MaxLista;
  GamaConj          = 0..PredTamañoConj;
  Criba             = SET OF GamaConj;
  Eratostenes       = ARRAY [dimension] OF Criba;
  cardinal          = 0..MaxInt;

VAR
  Cribas            : Eratostenes;
  contador,
  multiplo          : cardinal;
  indice            : dimension;
  N                 : GamaPrimo;
  numero            : 0..PredTamañoConj;

BEGIN
  WriteLn;
  WriteLn   ('Criba de Eratostenes': 50);
  WriteLn   ('============': 50);
  WriteLn;
  WriteLn;

  FOR indice := 0 TO MaxLista DO
    Cribas [indice] := [0..PredTamañoConj];
      {poner todos los numeros en las cribas}
  Cribas [0] := [2..PredTamañoConj];
  {WriteLn (1);} {numero primo por definicion}
  contador := 1;

  FOR N := 2 TO PrimoMax DO
    BEGIN
      indice := N DIV TamañoConj;
      numero := N MOD TamañoConj;

      IF numero IN Cribas [indice] THEN
        BEGIN
          contador := succ (contador);
        {WriteLn (N);}
          Cribas [indice] := Cribas [indice]
          - [numero];
          multiplo := N+N;

          WHILE multiplo <= PrimoMax DO
            BEGIN
              indice:=multiplo DIV TamañoConj;
              Cribas [indice] := Cribas [indice]
              -[multiplo MOD TamañoConj];
              multiplo := multiplo+N
            END
        END
    END;

  WriteLn;
  WriteLn (contador: 25,'primos hallados.')

END.
```

Robots en el C64

Usted está solo, abandonado en un planeta defendido por robots asesinos... Se trata de "Robots", en versión para el Commodore 64



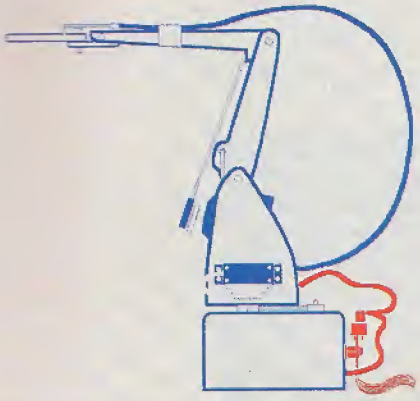
Las minas se hallan representadas sobre la pantalla mediante unas "X". Al comienzo del juego, cinco robots (representados por rombos) se hallan sobre el terreno. Sin perder un segundo, se precipitan hacia usted, siguiendo siempre el camino más corto. Por suerte, los robots son ciegos y no ven las minas que se hallan entre ellos y usted, lo cual le permite eliminarlos, siempre que se desplace del modo adecuado. Para ello debe usar las teclas:

Q, W, E, A, S, D, Z, X, C

según la dirección que haya escogido. La tecla S sirve para detenerse. Cuando haya eliminado a todos los robots, el juego continúa con un robot suplementario. Si salta sobre una de las minas o le mata uno de los robots, aún no se ha perdido todo. En realidad tiene cinco vidas. Si desea cambiar el número de minas, modifique el valor de la variable NM en la línea 1270.

```
  5 REM  ************
 10 REM *   ROBOTS   *
 15 REM  ************
 20 DIM R(30)
 25 NH=5
 30 GOSUB 1000
100 GET X$
110 GOSUB 900
120 J=J+D
130 C=PEEK(J)
140 IF C<>CN AND C<>CJ THEN 600
150 POKE J1,CN
160 POKE J,CJ
170 POKE J+M,JC
180 J1=J
200 JY=INT((J-1024)/40)
210 JX=(J-1024)-JY*40
220 T=0
230 FOR I=1 TO NR
240 IF R(I)=0 THEN 380
250 T=1
260 RY=INT((R(I)-1024)/40)
270 RX=(R(I)-1024)-RY*40
280 R1=RY+SGN(JY-RY)
290 R2=RX+SGN(JX-RX)
300 RR=40*R1+R2+1024
310 C=PEEK(RR)
320 IF C=CR OR C=CM THEN S=S+1:POKE R(I),
    CN:R(I)=0:GOTO 380
330 IF C=CJ THEN 600
340 POKE R(I),CN
350 POKE RR,CR
360 POKE RR+M,RC
370 R(I)=RR
380 NEXT I
390 IF T=0 THEN 500
400 FOR I=1 TO 500
410 NEXT I
420 GOTO 100
500 S=S+10
510 IF NR<30 THEN NR=NR+1
520 GOSUB 1030
530 GOTO 100
600 NH=NH-1
610 POKE J,CN
620 FOR I=1 TO 4
630 POKE 53281,0
640 FOR K=1 TO 50
650 NEXT K
660 POKE 53281,5
670 FOR K=1 TO 50
680 NEXT K
690 NEXT I
700 IF NH>0 THEN NZ=N1:GOTO 30
710 IF S>=RE THEN RE=S
720 PRINT CHR$(147)
730 FOR I=1 TO 4
740 PRINT
750 NEXT I
760 PRINT TAB(13)"PUNTOS[1SPC]:";
775 PRINT S
780 FOR I=1 TO 4
785 PRINT
790 NEXT I
795 PRINT TAB(11)"PUNTUACION[1SPC]MAXIMA [1SPC]:";
800 PRINT RE;
805 FOR I=1 TO 4
810 GET X$
815 PRINT
820 NEXT I
825 PRINT TAB(13)"OTRA[1SPC]?"
830 GET X$
835 IF X$="" THEN 830
840 IF X$<>"N" THEN S=0 : GOTO 25
845 END
900 IF X$="Q" THEN D=-41
910 IF X$="W" THEN D=-40
920 IF X$="E" THEN D=-39
930 IF X$="A" THEN D=-1
940 IF X$="S" THEN D=0
950 IF X$="D" THEN D=1
960 IF X$="Z" THEN D=39
970 IF X$="X" THEN D=40
980 IF X$="C" THEN D=41
990 RETURN
1000 CN=32
1010 N1=5
1020 NR=N1
1030 GOSUB 2360
1040 D=0
1050 CM=24
1060 CR=90
1070 CJ=81
1080 J=1024
1090 M=54272
1100 JC=0
1200 RC=0
1210 RY=0
1220 RX=0
1230 R1=0
1240 R2=0
1250 RR=0
1260 MC=0
1270 NM=60
1980 POKE 53280,6
1990 POKE 53281,6
2010 FOR I=0 TO 39
2020 POKE 1064+I,CM
2030 POKE 1064+M+I,MC
2040 POKE 1984+I,CM
2050 POKE 1984+M+I,MC
2060 NEXT I
2070 FOR I=1 TO 22
2080 POKE 1064+I*40,CM
2090 POKE 1064+M+I*40,MC
2100 POKE 1103+I*40,CM
2110 POKE 1103+M+I*40,MC
2120 NEXT I
2130 FOR I=1 TO NM
2140 GOSUB 3000
2150 POKE P,CM
2160 POKE P+M,MC
2170 NEXT I
2180 FOR I=1 TO NR
2190 GOSUB 3000
2200 R(I)=P
2210 POKE P,CR
2220 POKE P+M,RC
2230 NEXT I
2240 GOSUB 3000
2250 J=P
2260 POKE J,CJ
2270 POKE J+M,JC
2280 J1=J
2290 FOR I=1 TO 5
2300 POKE J,CJ+128
2310 FOR K=1 TO 100
2320 NEXT K
2330 POKE J,CJ
2340 FOR K=1 TO 100
2345 NEXT K
2350 NEXT I
2355 RETURN
2360 PRINT CHR$(144);
2370 PRINT CHR$(147);
2380 PRINT "PUNTOS[1SPC]:";S,
2390 FOR I=1 TO NH
2400 PRINT "H";
2410 NEXT I
2420 RETURN
3000 P=INT(RND(TI)*960)+1064
3010 IF PEEK(P)<>32 THEN 3000
3020 RETURN
```

Conexiones

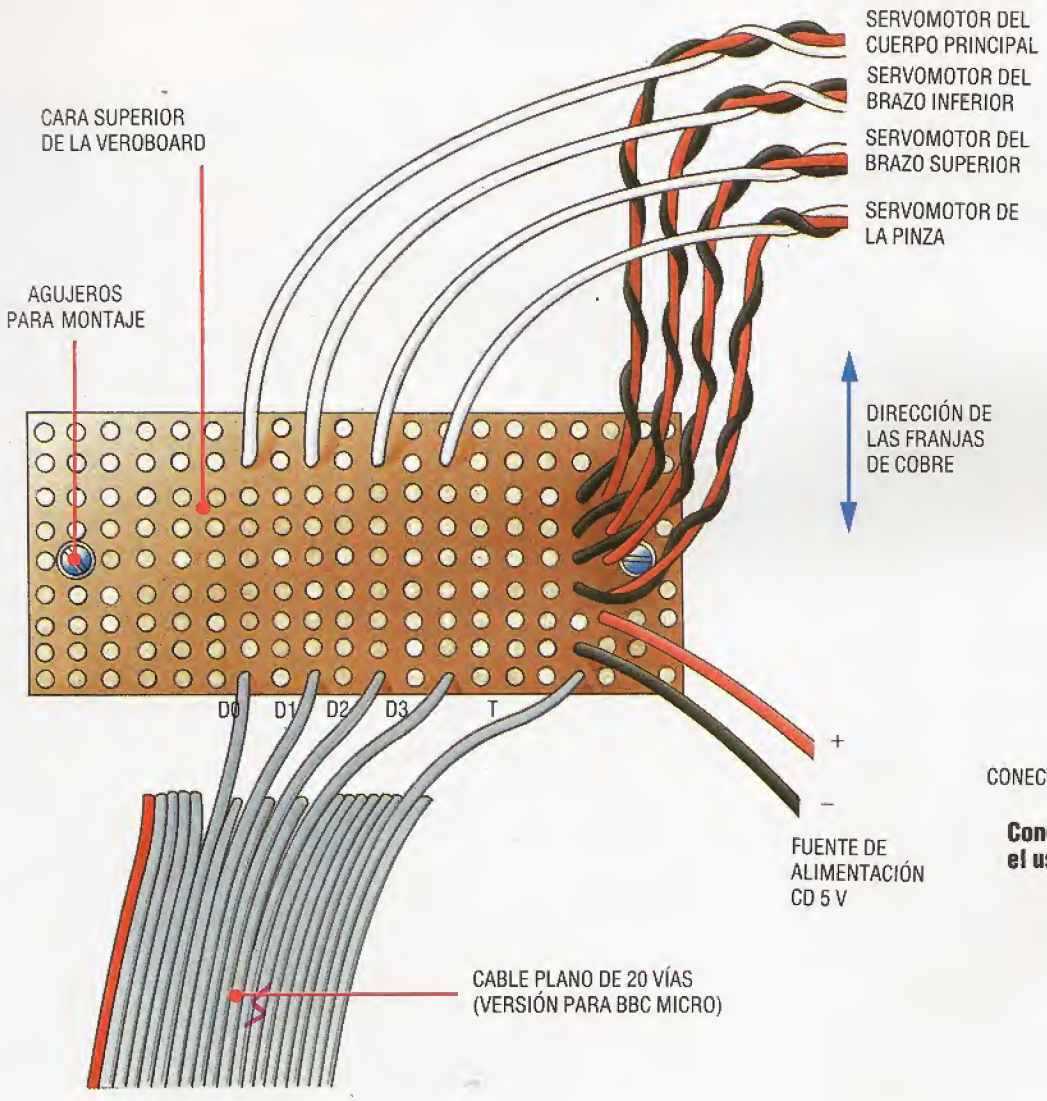


Habiendo cubierto ya las principales etapas de construcción de nuestro brazo-robot, sólo nos resta efectuar las conexiones eléctricas

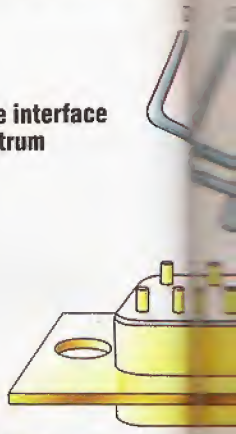
Paso 1: Conexiones de la veroboard

Cada servomotor tiene tres cables unidos a él. En los servomotores Futaba que recomendamos para este proyecto, los colores de estos cables son: blanco para la línea de control, rojo para la línea de potencia positiva y negro para la línea de retorno común. Los cables del servomotor se deben unir al ordenador controlador y a la fuente de alimentación utilizando un rectángulo de veroboard de 20 franjas ×9 aguj., que se montará en la parte posterior de la caja base del brazo. Corte la veroboard al tamaño adecuado y taladre dos agujeros para el montaje en las posiciones que se indican. Utilice la veroboard a modo de plantilla para marcar las posiciones de los agujeros en la parte posterior de la caja base y haga las perforaciones. Los agujeros de la caja base y la veroboard se deben taladrar de modo que a través de ellos se pueda colocar un tornillo para metales. Haga otro agujero en la parte posterior de la caja base y haga pasar a través del mismo el grupo de cables del servomotor montados dentro de la base. Una entre sí los cuatro grupos de cables de los servos a la parte posterior de la caja base y suelde los cables en las posiciones que se indican. Las cuatro líneas de datos y la línea a tierra del ordenador se deben soldar en la parte de abajo de la veroboard. El diagrama ilustra un cable plano de 20 vías que se utiliza con el BBC Micro. Para el Spectrum y el Commodore se emplean cinco

Paso 1: Conexiones de la veroboard

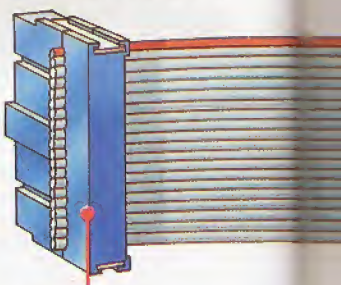


Paso 2: Conexiones



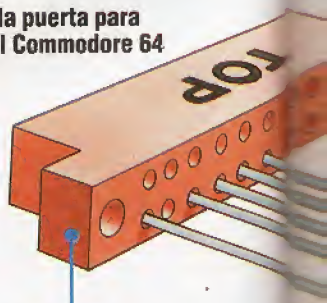
Adaptador de interface para el Spectrum

Conector de la puerta para el usuario del BBC Micro



CONECTOR IDC DE 20 VÍAS

Conector de la puerta para el usuario del Commodore 64



CONECTOR MARGINAL DE 24 VÍAS AL ORDENADOR

cables separados, o un trozo de cable plano de cinco vías. Observe que las líneas de control blancas de los servomotores se sueldan en posiciones que se hallan directamente encima de cada una de las líneas de datos del ordenador, y que todas las líneas negras se sueldan en una única franja de cobre, encima de la conexión a tierra del ordenador y las dos conexiones de la fuente de alimentación. La tarea final de soldadura consiste en fijarle a la placa una fuente de alimentación de 5 V, como se observa en la ilustración. La fuente de alimentación de 5 V de que disponen el BBC Micro, el Spectrum o el Commodore 64 no es apropiada para alimentar los cuatro servomotores con una carga pesada. Debería buscarse, entonces, una fuente alternativa. Una pila de 4,5 V (o tres pilas de 1,5 V unidas con un sujetador de pilas) constituye una fuente de alimentación ideal. Si dispone de un transformador CD de 5 V, también lo puede utilizar. Si pretende emplear pilas, entonces debe soldar un sujetador de pilas en los extremos libres de las líneas de potencia que se extienden desde la veroboard.

Si usted utiliza un transformador, debe instalar un conector de potencia en línea adecuado. El lado negativo de la fuente de alimentación comparte la misma franja de cobre con la conexión a tierra del ordenador y las líneas de retorno negras de los servomotores; el cable de potencia positivo se conecta a cada uno de los cables rojos de los servomotores a través de una franja de cobre común.

Paso 2: Conexiones puerta ordenador

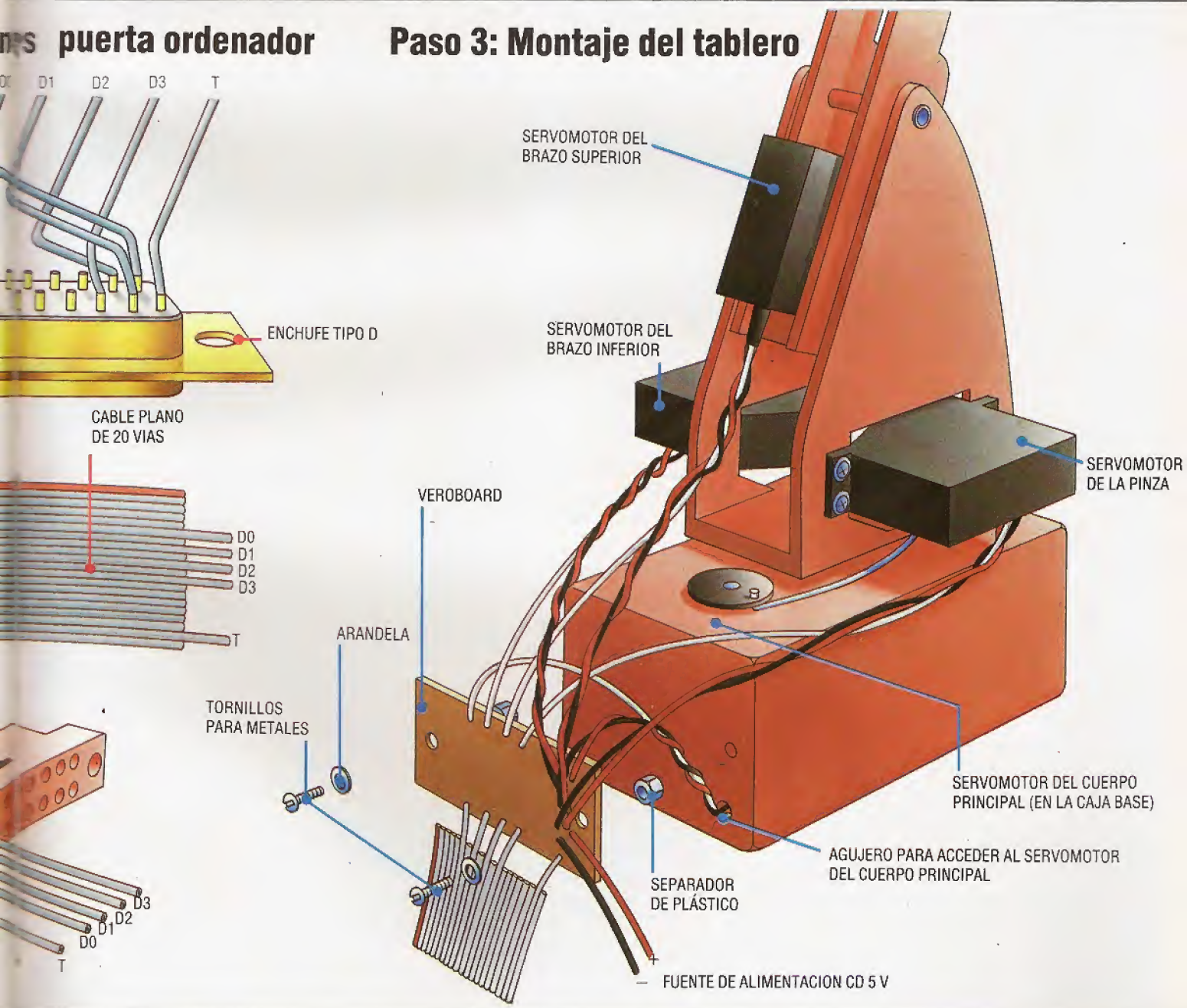
Después de realizar el cableado de la veroboard, debe añadirse el conector adecuado a la puerta del ordenador a los extremos libres de las líneas de datos de D0 a D3 y la línea de tierra. Para el BBC Micro debe utilizarse un cable de 20 vías estándar y un conector IDC. El Commodore 64 emplea un conector marginal de 0,15 pulgadas y 24 vías. Debido a que este conector se enchufa en la puerta para el usuario hacia arriba o hacia abajo indistintamente, antes de comenzar marque uno de los lados como ARRIBA. Las cinco líneas de la veroboard deben soldarse tal como se indica. Tanto el BBC Micro como el Commodore 64 poseen un sistema de circuitos incorporado para tratar las aplicaciones de control a través de una puerta para el usuario. El Spectrum, sin embargo, carece de un sistema de circuitos de este tipo, y debe construirse una interface especial para enchufar en su puerta de ampliación. En anteriores capítulos ya hemos diseñado y construido una interface de este tipo. Ahora los usuarios de un Spectrum habrán de remitirse a aquellos capítulos para construirse la interface. Un conector tipo D de 12 vías que se enchufa directamente en el robot proporciona las líneas de datos y de potencia de la interface. Podemos adaptar este conector para utilizarlo con el brazo-robot. Confeccione un enchufe tipo D de 15 vías empleando los extremos libres de las líneas de datos y tierra provenientes del trozo de veroboard del brazo, y añádale una cubierta para enchufe. Este adaptador permite enchufar el brazo-robot en la puerta para ampliación del Spectrum, a través de la placa de interface.

Paso 3: Montaje del tablero

Cuando haya terminado todo el cableado y lo haya probado cuidadosamente, debe montar la veroboard en la parte posterior de la caja base utilizando dos tornillos para metales ajustados mediante tuercas dentro de la base. Asegúrese de que los motores queden conectados a la veroboard en el orden correcto. La línea de datos D0 (la más alejada hacia la izquierda) debe conectarse a la línea blanca del servomotor del cuerpo principal, montado en la caja base; D1 debe conectarse al servomotor del brazo inferior, montado a la izquierda del cuerpo principal; D2 controla el servomotor del brazo superior, montado junto a la juntura del hombro, y D3 controla el servomotor de la pinza, montado a la derecha del cuerpo principal.

puerta ordenador

Paso 3: Montaje del tablero



Kevin Jones

HACIENDO REGISTROS

Muchos DBM poseen un lenguaje incorporado que permite efectuar búsquedas y acceder a la información con un mínimo esfuerzo

Las bases de datos no se limitan sólo a organizar datos como un conjunto de registros en un archivo. Es posible acceder a los registros mediante instrucciones sencillas, pero los mejores DBM permiten que los usuarios escriban procedimientos empleando un lenguaje de programación incorporado.

Para comprender el empleo en las bases de datos de procedimientos escritos por el usuario, debemos primero repasar la forma en que utilizan las instrucciones la mayoría de los DBM. Una instrucción, como podría ser SEARCH, LOCATE, DISPLAY, NEXT, LAST, etc., es procesada por el DBM para averiguar lo que desea el usuario. Habiendo hecho esto, abre el archivo de la base de datos y lo revisa secuencialmente, comenzando por el primer registro y recorriéndolo todo, hasta el último, extrayendo por el camino aquellos registros que satisfagan las exigencias. Se suele decir que tales instrucciones pertenecen a un "lenguaje de interrogación" (*query*).

En claro contraste con las instrucciones directas o interrogaciones, muchos DBM (dos buenos ejemplos de éstos son *Archive*, de Psion, y *dBase II*, de Ashton Tate) permiten que el usuario escriba programas o procedimientos para simplificar el proceso de extracción o confrontación de la información. Estos procedimientos no son meros conjuntos de instrucciones o interrogaciones estándares. Están escritos en lenguajes de programación completos (si bien limitados) que están incorporados como parte del DBM. Algunos de estos lenguajes de programación de DBM son bastante similares a los de programación normales, y son bastante fáciles de aprender. El *Archive* incorpora un lenguaje que es idéntico al SuperBASIC de Sinclair. Escribir procedimientos para utilizar con *Archive* es tarea sencilla para quien esté familiarizado con el SuperBASIC.

Si bien el lenguaje dBase II no se parece exactamente a ningún otro lenguaje, guarda suficiente parecido a un BASIC estructurado con unas pocas palabras reservadas fáciles de aprender y utilizar. Sus estructuras son bastante sencillas, e incluyen DO WHILE...ENDDO, DO CASE...ENDCASE e IF...ENDIF. Además hay alrededor de un centenar de instrucciones, algunas familiares y otras hechas a medida específicamente para emplear en un entorno de base de datos. Las instrucciones familiares incluyen CALL (para llamar a código máquina), NOTE (equivalente al REM del BASIC), STORE<expresion >to<variable> (equivalente al LET<variable >=<expresion> del BASIC) y muchas otras. Las instrucciones menos familiares incluyen SKIP (para ir hacia adelante o hacia atrás a través de los registros del archivo), PACK (para suprimir del archivo registros no deseados) y FIND<serie de caracteres>.

También se proporcionan otras varias funciones, incluyendo algunas familiares similares al BASIC y otras más inusuales. Ejemplos de ellas son CHR<expresion> (equivalente a CHR$(x)), LEN<expresion de serie de caracteres> (equivalente al LEN del BASIC), TYPE<expresion> (que devuelve el tipo de expresión) y DATE() (una variable del sistema que contiene la fecha).

Para apreciar lo útil que puede ser un procedimiento escrito a la medida, en comparación con el simple uso de una secuencia de instrucciones desde el teclado para extraer la información, emplearemos *dBase II* para crear una base de datos de stock e inventario, y luego escribiremos un procedimiento sencillo en lenguaje dBase II para extraer cierta información. Los registros del archivo serán bastante simples, con cuatro campos numéricos y un campo de caracteres. Un registro típico (también utilizado en el capítulo anterior) sería así:

```
NUMERO COMPONENTEFABRICANTE          06116
NUMERO COMPONENTENUESTRO              3995
CANTIDAD EN STOCK                       86
PRECIO                                34.75
DESCRIPCION          Puerta delantera SEAT 127
```

Primero hemos de hallar un nombre de una sola palabra para cada campo y decidir las longitudes de los campos y si han de ser campos *de caracteres*, *numéricos* o *lógicos*. Esto sería suficiente:

```
COMPONENTEFABRICANTE : 5 carac.; numérico
COMPONENTENUESTRO    : 5 carac.; numérico
EXISTENCIAS          : 3 carac.; numérico
PRECIO               : 6 carac.; numérico
DESCRIPCION          :40 carac.; de caracteres
```

Una vez puesto en funcionamiento, *dBase II* responde con un punto de requerimiento en la pantalla que indica que está esperando una entrada. La instrucción para iniciar un nuevo archivo es CREATE; también deberemos proporcionar un nombre para el archivo en su conjunto (utilizaremos COMPONENTES como nombre del archivo). El proceso sería algo así:

```
.create componentes <CR>
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD  NAME, TYPE, WIDTH, DECIMAL PLACES
(entrar estructura del registro del siguiente modo:
campo nombre, tipo, anchura, posiciones decimales)
001    componentefabricante,n,5,0
002    componentenuestro,n,5,0
003    existencias,n,3,0
004    precio,n,6,2
005    descripcion,c,40
006    <CR>
```

Nuestra entrada se ha dado en letras minúsculas y la respuesta de *dBase II* se ha impreso en letras mayúsculas. Observe que los campos numéricos requieren que, además de la anchura del campo, se especifique el número de posiciones decimales. Pulsando la tecla Return sin entrar ninguna información, como en el campo 006, se termina la fase inicial de creación de un archivo.

A continuación hemos de entrar algunos datos. Esto se realiza utilizando la instrucción APPEND. Ésta hace que se limpie la pantalla y aparezca un "esqueleto" del registro, a la espera de que se entren datos.

```
COMPONENTEFABRICANTE :
COMPONENTENUESTRO    :
EXISTENCIAS          :
PRECIO               :
DESCRIPCION          :
```

Ahora se pueden digitar los datos en el teclado, y se los pasará a cada campo. Un retorno de carro señala que la entrada de un campo ha terminado; apenas se pulse Return para el campo final, se aceptarán todos los datos y volverá a aparecer el esqueleto del registro, esperando detalles para el siguiente registro. Para terminar la entrada de datos se pulsa Return al comienzo de un registro vacío.

Habiendo entrado los datos en el archivo, usted deseará utilizarlos. Supongamos que, para efectuar una verificación contable, necesitara saber el valor de venta al por menor de todo su stock. Una forma en que se podría hacer esto sería ir mirando cada registro de uno en uno en la pantalla y tomando nota de la cantidad de componentes que hay en stock de cada artículo, con el precio de cada uno, y multiplicando las dos cifras. Tras examinar todos los registros y apuntar los precios y cantidades de cada artículo del stock, podría obtener el valor total multiplicando cada precio por el número de componentes y sumando después todos los subtotales. Si estuviera utilizando un fichero de tarjetas convencional, ésta sería la única forma de llegar a la solución.

La mayoría de la DBM permiten recuperar esta clase de información de modo mucho más sencillo. He aquí cómo se podría hacer con *dBase II*:

```
.use componentes
.sum existencias*precio
37870.58
```

La primera línea es una instrucción que solicita a *dBase II* que trabaje con el archivo llamado COMPONENTES. La segunda línea es una instrucción que significa "sumar entre sí los resultados obtenidos tras multiplicar el campo EXISTENCIAS por el campo PRECIO de cada uno de los registros". La tercera línea es la clase de respuesta que podría devolver para el valor total de todo el stock. Muchísimo mejor que los ficheros de tarjetas, ¿verdad?

Este ejemplo ilustra claramente cuán eficaz puede ser un buen DBM, incluso aunque no se utilice más que una sola instrucción. Pero su verdadera eficacia se observa al almacenar secuencias de instrucciones en forma de programa. Para almacenar nuestras sencillas instrucciones de valoración del stock en la forma de un programa denominado VALOR, se necesitaría el siguiente diálogo:

```
.modify command
ENTER FILE NAME: valor
set talk off
use componentes
sum existencias*precio
set talk on
cancel
```

Este corto programa se almacenará en disco y podrá ser empleado desde *dBase II* en cualquier momento digitando la instrucción:

```
.do valor
```

El programa se leerá y se ejecutará automáticamente y, apenas concluido, el control retornará a *dBase II*.

Procedimiento contable

```
*Visualizar contables
SET TALK OFF
USE SOCIOS
DO WHILE.NOT.EOF
  IF PROFESION="CONTABLE"
    DISPLAY NOMBRE
  ENDIF
  SKIP
ENDDO
```

El programa que ofrecemos, que extrae los nombres de los socios de un club que sean contables, está escrito en el lenguaje de instrucciones dBase II de Ashton Tate. La primera línea, que comienza con un asterisco, indica que se trata de una línea de comentarios. No requerimos que se visualicen todos los números de registros mientras *dBase II* va efectuando la búsqueda, de modo que interrumpimos la conversación (SET TALK OFF). En la base de datos podría haber retenidos varios archivos diferentes, de modo que debemos especificar en qué archivo (en este caso, SOCIOS) se ha de llevar a cabo la búsqueda. Puesto que deseamos buscar a través de todos los registros disponibles, construimos un bucle alrededor de la estructura DO WHILE...ENDDO. Dentro del bucle establecemos una condición que afirma que siempre que el campo de la profesión contenga la serie "CONTABLE", el programa debe visualizar el campo del nombre del registro. Observe que la condición IF debe concluir con un ENDIF. El programa ejecuta entonces SKIP, que le indica a *dBase II* que pase al siguiente registro

Descubriendo nuevos mundos
El Discovery 1 está considerado como un sistema de ampliación "todo en uno" para el usuario del Spectrum. La máquina viene completa con una unidad de disco, sistema operativo y conexiones que permiten la instalación de impresoras, palancas de mando, monitores de video compuesto y otros periféricos, sin necesidad de interfaces adicionales

Sistema total

Con el examen del Discovery 1, de Opus, finalizamos nuestra serie dedicada a las alternativas al microdrive de Sinclair

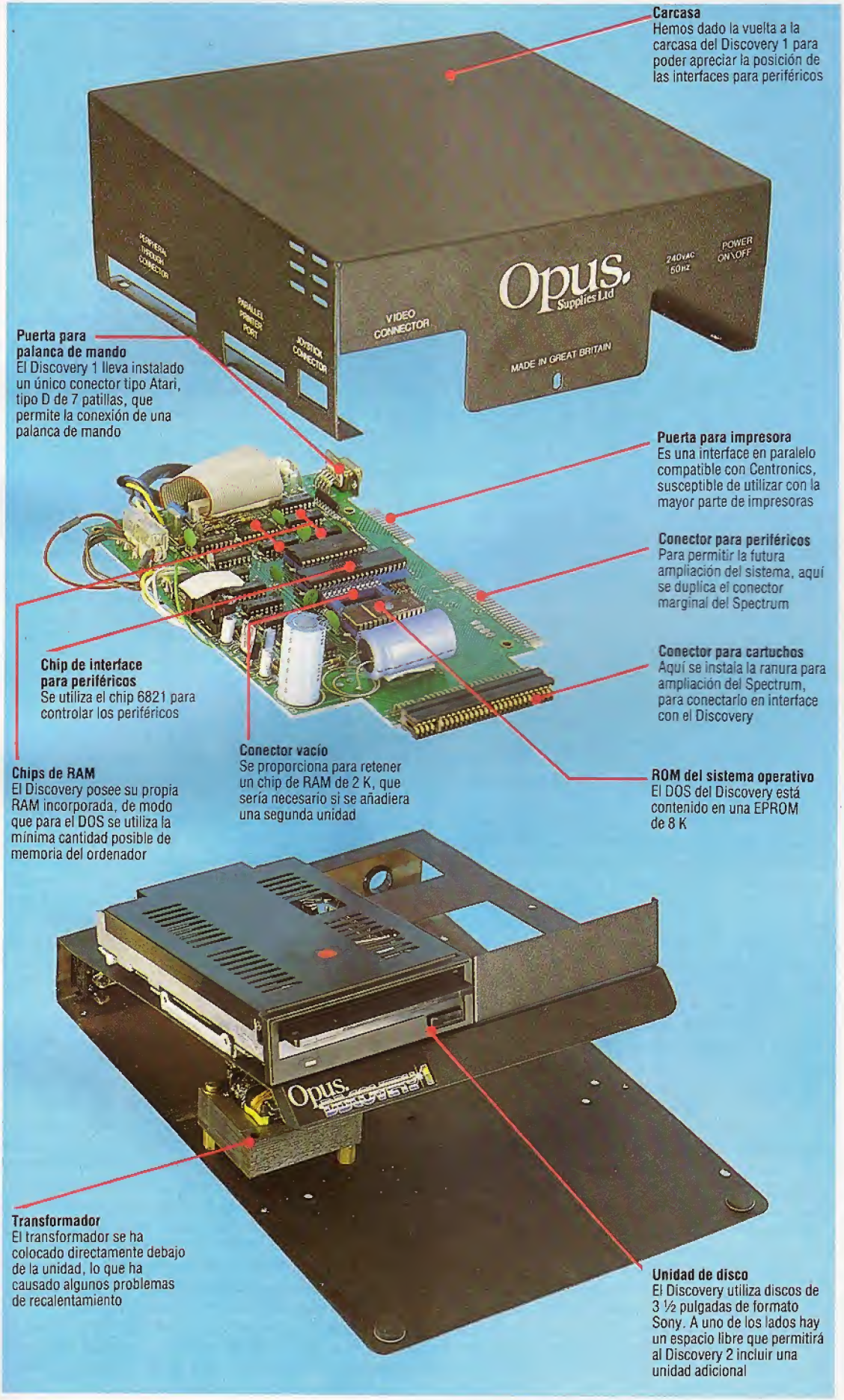
En el capítulo anterior de este apartado analizamos el Wafadrive de Rotronics. Si bien este dispositivo resultó ser algo más fiable que el microdrive, en comparación es más lento y basado todavía en el sistema, más bien sospechoso, de cinta continua. En este capítulo centramos nuestra atención en un enfoque más convencional al almacenamiento masivo: un sistema basado en disco de Opus Supplies.

Una unidad de disco para el Spectrum no constituye ninguna idea nueva. Durante los dos últimos años han aparecido diversos sistemas operativos de disco e interfaces para la máquina; sin embargo, ninguno de estos sistemas se ha hecho especialmente popular. Ello se debe en parte a que las interfaces se han promocionado sólo en la prensa especializada, lo que puede haber dado la impresión de que los dispositivos estaban destinados sólo al entusiasta del Spectrum y a los programadores de lenguaje máquina, y sólo se han puesto a la venta a través de casas de venta por correspondencia y no se les ha dado el tipo de comercialización masiva necesaria para introducirlos en el público.

El Discovery 1 viene en una carcasa de chapa metálica, con un frente que se prolonga hacia adelante para soportar al Spectrum, que se enchufa en una ranura para cartuchos a través de la puerta para ampliación. Aunque la operación de este sistema parece comparativamente sencilla, quizá los usuarios encuentren dificultades para instalar el Discovery en el conector marginal. Ello se debe a que los cables de la cassette y la antena tienden a entorpecer la operación dificultando la correcta fijación en la ranura para cartuchos. Éste es un problema grave si se considera que un conector marginal colocado incorrectamente podría dañar gravemente al Spectrum. La dificultad no es tan pronunciada como con el Spectrum original (para el cual se diseñó originalmente el Discovery), pero el nuevo Spectrum+, con su carcasa más grande, puede exigir que los usuarios se esfuercen durante varios minutos antes de quedar satisfechos con la instalación correcta del dispositivo. Una vez que el Spectrum se ha instalado correctamente en el Discovery 1, la máquina en realidad bloquea la propia entrada de la fuente de alimentación del ordenador. El Discovery, por consiguiente, ha sido diseñado para alimentarse a sí mismo y al Spectrum, haciendo innecesaria la fuente de alimentación externa del micro.

Encima de la ranura para cartuchos de la izquierda hay una unidad de disco simple de 3 ½ pulgadas, con espacio a la derecha para una segunda unidad. (Opus tiene la intención de lanzar una versión de la máquina con unidad doble, denominada Discovery 2.) Los usuarios del Discovery 1 que deseen mejorar su sistema convirtiéndolo en un sistema de doble unidad, pueden aguardar el lanzamiento de una unidad de disco adicional que se llamará Discovery+. No obstante, los usuarios no se verán limitados sólo a las unidades de Opus, dado que la empresa afirma que también se pueden añadir unidades de 5 ¼ pulgadas.

Al igual que Rotronics, la filosofía de Opus al diseñar la máquina es la de proporcionar no sólo un sistema de almacenamiento masivo para el Spectrum, sino también añadir interfaces extras para periféricos que permitan a los usuarios el empleo de impresoras y otros dispositivos. En la parte poste-



DISCOVERY 1

DIMENSIONES
300×210×75 mm

INTERFACES
Conector paralelo, interface en paralelo Centronics, puerta para palanca de mando, enchufe para video compuesto

FORMATO
Discos estándares Sony de 3 ½ pulgadas, de doble densidad y doble cara

CAPACIDAD
250 K total, 180 K tras el formateado

VELOCIDAD
15 Kbaudios de velocidad de transferencia; tiempo de acceso de pista a pista: 3 milisegundos

rior del Discovery 1 se proporciona un conector para monitor de video compuesto, según declara un portavoz de Opus, para los usuarios de gestión que deseen incorporar una pantalla monocromática (aunque, por supuesto, el video compuesto sí produce una señal de color) para prolongados períodos de tratamiento de textos. No obstante, es de lamentar que en una máquina que destaca por sus coloridos programas de juegos Opus no haya podido proporcionar una interface RGB para producir una imagen mucho más nítida.

En el lado derecho del Discovery hay una puerta para una palanca de mando tipo Atari compatible con Kempston, junto a la cual hay una puerta para impresora en paralelo Centronics bidireccional. Por último, hay un conector para periféricos que permite la conexión de otras interfaces compatibles con el Spectrum, como un monitor RGB.

Al igual que en el Wafadrive, el sistema operativo de disco del Discovery es muy similar al de la Interface 1; por ejemplo, para generar una instrucción se requiere <INSTRUCCIÓN>*. Cuando el intérprete de BASIC llega al *, no lo reconoce como si se tratase de una instrucción de BASIC e intenta generar un mensaje de error de sintaxis. No obstante, el DOS lo intercepta y lo envía a su propio sistema operativo de 8 K en la posición de los 8 K inferiores de la ROM del Spectrum, y luego interpreta la instrucción. Hay que destacar que si el usuario hubiera realizado un error de sintaxis, el DOS no lo reconocería y se generaría un mensaje de error, si bien ello se haría a través de la ROM del DOS.

Al diseñar su sistema DOS Opus ha ido más lejos que Rotronics al proporcionar compatibilidad: se han conservado todas las instrucciones disponibles en el microdrive. Esto obedece a varios motivos. Debido al sistema de entrada del Spectrum, de una única palabra clave, obviamente es más fácil escribir un sistema operativo que utilice instrucciones inherentes, en vez de meterse en el problema de escribir instrucciones propias. Esto también significa que los usuarios que ya estén familiarizados con el sistema operativo del microdrive podrán utilizar el Discovery de inmediato, puesto que toda la sintaxis es la misma. Además, trabajar con un sistema operativo puede llevar a todo tipo de problemas imprevistos con el mapa de memoria. Ello significa que los programas compatibles con la Interface 1 podrían no ser necesariamente compatibles con su sistema operativo remendado, un problema que se ha presentado con asiduidad en muchos periféricos de otras empresas independientes.

La forma en que Opus ha seguido de cerca el sistema de instrucciones del microdrive de Sinclair se hace más evidente cuando se observa la manera en que se han organizado las corrientes. En el Spectrum, los canales de salida están organizados en 16 corrientes numeradas de la 0 a la 15. Tres de éstas están reservadas para la pantalla, el teclado y la impresora, y las otras están libres para utilizarlas con cualquier otro periférico. En la lista de instrucciones de Sinclair para la Interface 1 existen diversos caracteres individuales que abren canales a dispositivos específicos; por ejemplo, 'm' para microdrive. El Discovery ha adaptado las mismas para su propio uso, de modo que la instrucción LOAD*'m';1;'nombre' funcionará igualmente en el Discovery 1 y en el microdrive, aunque en este caso 'm' alude a la unidad de disco. Sin embargo, por razones de conveniencia, Opus ha adaptado el formato de instrucción de modo que se pueda omitir la 'm', abreviando de este modo el sistema de Sinclair, en cierta forma muy largo. Asimismo, hay otras instrucciones que también se han adaptado. En el microdrive, el carácter 't' en una instrucción abre un canal a la interface RS232, mientras que en el Discovery el mismo carácter abre el canal a la impresora en paralelo.

Los discos en funcionamiento

La unidad de disco que se ofrece con el sistema emplea los discos de 3 ½ pulgadas de formato Sony de doble densidad, que cada día se utilizan con mayor asiduidad en los microordenadores. Cada uno de los discos posee una capacidad total de 180 K de espacio de almacenamiento. El DOS soporta acceso directo cuando se busca en un archivo, lo acceso directo cuando se busca en un archivo, lo que es considerablemente más rápido que los métodos de búsqueda secuencial que utilizan algunos otros sistemas de disco. Asimismo, no existe límite alguno en cuanto a la cantidad de archivos que se pueden retener en un disco, lo que puede ser importante cuando se desea guardar numerosos archivos cortos. Si el directorio se llenara rápidamente, podría quedar en el disco una gran cantidad de espacio que no se podría aprovechar.

Al comparar el tiempo que lleva guardar (SAVE) y cargar (LOAD) un archivo utilizando el Discovery y el microdrive, el primero resultó algo más rápido para encontrar el archivo, pero considerablemente más lento para guardarlo y cargarlo. Hallar un archivo es más rápido en el sistema de disco porque los archivos están organizados por acceso directo, mientras que los microdrives, por su propia naturaleza, son dispositivos de acceso secuencial. Las razones por las cuales cargar un archivo en la memoria es mucho más lento resultan difíciles de explicar, pero la realidad es que la velocidad de transferencia del Discovery 1 es inferior a la de los microdrives: 15 Kbaudios frente a 19,2 Kbaudios. La verdadera ventaja del Discovery radica en tener un sistema de almacenamiento masivo más eficaz que el del microdrive, y un medio de almacenamiento que dispone de una gama de fabricantes más amplia.

Opus parece haber pensado muy bien en el problemático aspecto del apoyo de software para el Discovery. Obviamente, contar con una gran empresa que venda el producto en su cadena de establecimientos es una ventaja, porque las empresas de software pueden ofrecer sus productos junto con la propia máquina. La empresa también ha señalado que muchas casas de software ya se han comprometido a transferir algunos de sus programas existentes a los discos de 3 ½ pulgadas.

El lanzamiento de la serie de unidades de disco Discovery ha sido evidentemente muy bien planeado y Opus, como es lógico, ha intentado proporcionar a su nueva línea las mayores probabilidades de éxito posibles. La tarea primordial que tiene la empresa ante sí es convencer a los usuarios del Spectrum de que el Discovery constituye una inversión más valiosa para sus máquinas que la que representa la alternativa de Sinclair.

Llaman a la puerta

Vamos a realizar un estudio de las entradas y salidas tratadas por el sistema operativo del Commodore 64

La fotografía que adjuntamos muestra todas las puertas disponibles en la parte trasera del Commodore 64. Junto a la puerta para cassette, para audio/video y para la TV, existen tres puertas de E/S a disposición del programador. Son, de izquierda a derecha, la puerta para ampliación (también denominada puerta para cartucho) la puerta serial (o bus serial) y la puerta para el usuario. Veámoslas por separado.

- *Puerta serial*: Las impresoras seriales Commodore y la unidad de disco serial 1541 se conectan mediante el enchufe DIN de seis patillas. La instrucción OPEN hará siempre referencia a esta puerta con cualquier número de dispositivo salvo el 2 (especificar el dispositivo 2 significa la RS232 por medio de la puerta para el usuario). Por ejemplo, con el número de dispositivo 8, la instrucción OPEN2, 8,2,"NOMBREARCHIVO" abrirá un archivo en la unidad de disco y a través de la puerta serial. No es aconsejable hacer uso de la puerta serial por medio de rutinas en BASIC o del núcleo, salvo con dispositivos Commodore.

Se pueden adquirir interfaces que aceptan mensajes seriales a través de esta puerta y realizan una conversión paralelo serial/IEEE. Esto permite que los dispositivos periféricos sean conectados al PET Commodore y puedan ser empleados en el Commodore 64. Tales periféricos incluyen las unidades de disco Commodore 4040.

- *Puerta para el usuario*: Es un conector plano de 24 vías, macho, de 0,15 pulgadas, utilizable tanto para la comunicación en serie como en paralelo. Por ejemplo, esta puerta puede servir para conectar el Commodore 64 a una impresora no Commodore (como la Epson, que es la más frecuente), la cual es tratada como un dispositivo RS232. Ya analizaremos más adelante cómo se puede manipular la RS232 desde el OS.

La puerta para el usuario puede también emplearse en la comunicación en paralelo de 8 bits, pero ha de escribirse (o adquirirse) el software específico para gestionar dispositivos, ya que el OS no posee rutinas de control, o *drivers* (código máquina para control de periféricos), para la comunicación en paralelo. En el próximo capítulo ofreceremos un programa titulado Parawedge, que ilustra el trabajo global necesario para obtener un programa de comunicaciones en paralelo por medio de interrupciones con un grado de sofisticación mediano. Este programa podrá servir para transferir un bloque de memoria desde el ordenador a algún dispositivo externo (p. ej., trasladar un programa en BASIC de un Commodore 64 a otro), siempre que ambas máquinas tengan niveles de 5 V.

- *Puerta para ampliación*: Se trata de una entrada hembra de 44 patillas que proporciona el acceso a todas las líneas de control principal, dirección y datos del Commodore 64. Esta puerta sirve para acoplar cartuchos de juegos y cartuchos de IEEE en paralelo, responsables de la conexión del ordenador con periféricos PET. Sirve igualmente para cualquier otro dispositivo o software que requiera casi un completo control de la máquina o programas suficientemente aptos para garantizar el diseño y la construcción de una tarjeta para retenerlos como ROM o EPROM.

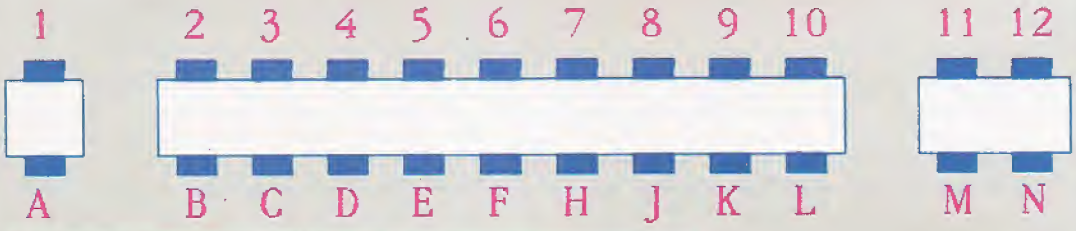
Proporcionamos unos diagramas de cada una de estas tres puertas mostrando las funciones de sus conexiones de patillas. Volvamos ahora a ocuparnos de aspectos del empleo que hace el sistema operativo de estas puertas. El conjunto de rutinas OS que se encargan de E/S se conoce como núcleo. Son programas en código máquina invocados por las sentencias en BASIC de E/S tales como OPEN, CLOSE, GET#, PRINT#. La organización de las rutinas del núcleo es tal que el programador puede acceder a ellas con facilidad a través de sistemas en lenguaje máquina. Damos un breve listado del lenguaje máquina que permite el uso de la rutina del núcleo LOAD.

Comunicación en paralelo

También vamos a examinar aquí la capacidad RS232 incorporada en el Commodore 64. Consideraremos además cómo se usa, por ejemplo, para tratar un modem por medio de las rutinas RS232.

Pero no siempre será posible el empleo de rutinas de sistema para la E/S. A veces puede ser necesario el empleo de una comunicación en paralelo y grandes cantidades de datos para un dispositivo propio cercano, más que una comunicación en serie con algún dispositivo remoto. No existen rutinas de sistema para el manejo de la puerta para el usuario en el Commodore 64. Por ello la comunicación en paralelo implica la programación de los dos Adaptadores de Interface Complejos 6526 (CIA). Para iniciarle en ello, daremos en el próximo capítulo una rutina en ensamblador de transferencia de 8 bits de datos en paralelo. Tal rutina permite al ordenador leer o enviar datos por medio de la puerta para el usuario mientras se procesa al mismo tiempo un programa en BASIC. El malabarismo del tiempo compartido se logra haciendo que el código que lee o escribe datos acepte interrupciones. En el capítulo anterior de esta serie dimos un ejemplo de una cuña IRQ (requerimiento de interrupción). Aquí la rutina acepta una NMI ya que la línea de flag de la puerta para el usuario puede servir para generar una NMI (interrupción no enmascarable): es la segunda línea de interrupción para el procesador 6510. A diferencia de la IRQ, no se puede enmascarar una señal de interrupción en la línea NMI por medio de las instrucciones SEI y CLI.

Puertas de E/S en el Commodore 64



Parte superior

Patilla	Tipo	
1	GND	Tierra
2	—	+5 voltios a un máximo de 100 mA
3	—	RESET del sistema
4	CNT1	Contador de puerta serial CIA#1
5	SP1	Puerta serial CIA#1
6	CNT2	Contador de puerta serial CIA#2
7	SP2	Puerta serial CIA#2
8	PC2	Línea "apretón de manos" desde CIA#2
9	—	Esta línea se conecta con ATN en la puerta serial
10	—	Fase + de 9 V ac
11	—	Fase – de 9 V ac
12	—	Tierra

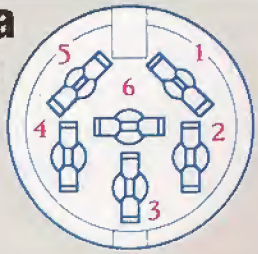
Notas:

1) Línea-3=protocolo X on/X off
Línea-X=protocolo CTS/RTS
2) (*) Además de ser direccionadas por medio de las rutinas del núcleo RS232 del OS, estas líneas pueden también programarse directamente para E/S definidas por el usuario. Tales conexiones se emplearon en el apartado *Bricolaje* para controlar dispositivos externos como la caja buffer y el robot móvil

Parte inferior

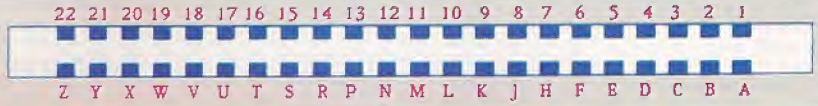
Patilla	Tipo	Función RS232 (*)	Usadas en línea-3/línea-X
A	GND	Tierra protectora	Ambas
B	FLAG2	Datos recibidos —entrada	Ambas
C	PB0	Datos recibidos —entrada	Ambas
D	PB1	Solicitud de envío (RTS) —salida	Ambas (alto en 3-1)
E	PB2	Terminal datos preparado (DTR) —salida	Ambas (alto en 3-1)
F	PB3	Indicador de timbre (RI) —entrada	—
H	PB4	Señal línea recibida (DCD) —entrada	Sólo línea-X
J	PB5	Sin asignar	—
K	PB6	Borrar para envío (CTS) —entrada	Sólo línea-X
L	PB7	Total datos preparado (DSR) —entrada	Sólo línea-X
M	Datos	Datos transmitidos —salida	Ambas
N	GND	Señal tierra	Ambas

Puerta serial



Patilla	Tipo	Observaciones
1	$\overline{\text{SRQIN}}$ Serial	Servicio de solicitud
2	GND	
3	ATN E/S SERIAL	Señal de atención para disp. en puerta
4	CLK E/S SERIAL	Reloj temporizador para puerta serial
5	E/S DATOS SERIAL	Línea de transporte datos de un solo bit
6	$\overline{\text{RESET}}$	Línea de RESET del hardware

Puerta ampliación



Parte superior

Patilla	Tipo	Observaciones
1	GND	
2	+ 5V	
3	+ 5V	
4	$\overline{\text{IRQ}}$	Petición de interrupción
5	R/$\overline{\text{W}}$	Posibilitar lectura/escritura
6	Reloj puntos	
7	E/S 1	
8	$\overline{\text{GAME}}$	
9	$\overline{\text{EXROM}}$	
10	E/S 2	
11	ROML	
12	BA	
13	$\overline{\text{DMA}}$	
14	D7	
15	D6	
16	D5	
17	D4	8 líneas de datos
18	D3	
19	D2	
20	D1	
21	D0	
22	GND	

Parte inferior

Patilla	Tipo	Observaciones
A	GND	
B	$\overline{\text{ROMH}}$	
C	$\overline{\text{RESET}}$	Línea de RESET del hardware
D	$\overline{\text{NMI}}$	Interr. no enmascarable
E	S02	
F	A15	
H	A14	
J	A13	
K	A12	
L	A11	
M	A10	
N	A9	
P	A8	16 líneas de dirección
R	A7	
S	A6	
T	A5	
U	A4	
V	A3	
W	A2	
X	A1	
Y	A0	
Z	GND	

El núcleo en acción

El núcleo (*kernel*) es una biblioteca de rutinas de E/S, que son llamadas por el BASIC o por código máquina escrito por el usuario. La ROM del núcleo está situada de $E000 a $FFFF, pero las rutinas deseadas se llaman desde una "tabla de salto", es decir, desde una tabla de punteros que apuntan a las posiciones de inicio de las rutinas y que se guarda en la parte superior de la memoria. La ventaja de este sistema de llamadas a través de una tabla de salto consiste en la facilidad con que puede emplearse el código máquina incluso con diferentes máquinas Commodore, pues la tabla de salto no varía aunque varíen los contenidos.

Para usar los programas núcleo se necesita material de referencia. La Guía de Referencia para el Programador que proporciona Commodore es quizá la mejor fuente. P. ej., para servirnos de la rutina de núcleo LOAD y realizar con ella un LOAD de reubicación, suele emplearse el siguiente fragmento de código. Primero se colocarán los códigos ASCII (según la versión del Commodore) correspondientes al nombre del archivo en direcciones de memoria consecutivas, y después se hará uso del siguiente código máquina:

```
LDA  #$01    Número lógico del archivo (lfn)
LDX  #$08    Número del dispositivo (disk)
LDY  #$00    Dirección secundaria (#$00 da
             la carga de reubicación)
JSR  $FFBA   Rutina de núcleo SLFS
             (establece lfn y dir. sec.)
LDA  #$0A    Long. nombre archivo
LDX  PLO     (p. ej., 10)
             Byte inferior del puntero a la dir.
LDY  PHI     de inicio del nombre archivo
             Byte sup. del puntero a la dir.
JSR  $FFBD   de inicio del nombre archivo
             Rutina de núcleo SETNAM
             (establecer nombre de archivo)
LDA  #$00    Carga=#$00/Verificación=#$01
LDX  DLO     Byte inf. de la dir. inicio destino
LDY  DHI     Byte sup. de la dir. inicio destino
JSR  $FFD5   Rutina de núcleo LOAD
```

Empleo de la RS232 en el Commodore 64

Siempre que se realice un planteamiento metódico no hay peligros serios en el empleo de rutinas OS para manipular la RS232, ya sea desde el BASIC o desde código máquina. Varios son los aspectos a considerar al aprestarnos a usar la RS232 a través de la puerta para el usuario. Dejando a un lado la velocidad en baudios, de la que volveremos a hablar, los únicos problemas con que se encontrará serán los referidos a estos puntos:

- El Commodore 64 funciona a niveles de 0 V a 5 V, mientras que la RS232 normal necesita niveles de −12 V a 12 V. Luego, a menos que no se comunique con otro Commodore 64, será preciso construir o adquirir un dispositivo para "conversión de niveles". Commodore proporciona un cartucho RS232 para esta circunstancia.
- Los códigos ASCII del Commodore se alejan de los códigos ASCII habituales, por lo que habremos de establecer dos tablas conversoras, una para transmitir y otra para recibir.

Siempre que esté abierto un canal RS232, el OS realiza un borrado automático de registros (CLR). Se pierden así inmediatamente todos los valores de las variables usadas por el BASIC con anterioridad; las instrucciones GOSUB darán mensaje de error al llegar a RETURN. Esto se debe a que la rutina de núcleo de la RS232 ocupa dos buffers de 256 bytes en la parte superior de la memoria. Esto puede ser problemático, ya que si no hay espacio suficiente para los buffers en esa zona, se corromperá el programa y no se tendrán mensajes de error.

- Si el programa en BASIC es largo o contiene gran número de asignaciones alfanuméricas, tarde o temprano obtendrá basura. Pueden perderse datos introducidos. Pero no siempre ha de suceder esto si el programa es de una longitud razonable y no se hacen tantas asignaciones alfanuméricas. Para abrir (OPEN) un canal RS232 desde el BASIC se emplea el siguiente formato:

```
OPEN 2,2,3,CHR$(CTRL)+CHR$(COMM)
```

CTRL y COMM son los bytes de control e instrucción, que contienen la información necesaria para establecer el canal. Nótese que CTRL y COMM deben ser dos bytes literales (o los PEEK de dos posiciones previamente llenadas) y no dos variables. Cada bit dentro de los bytes de CTRL y COMM tiene una función precisa (ver cuadros).

Por ejemplo, para abrir un canal RS232 con un bit de parada, una longitud de palabra de 7 bits, 300 baudios (total del byte de CTRL=38), paridad par, *full duplex* y control de flujo línea-3 (total del byte de COMM=96), emplearemos la instrucción:

```
OPEN2,2,3,CHR$(38)+CHR$(96)
```

Al decidir la velocidad de transmisión no se han de olvidar los siguientes factores. Al enviar información, la velocidad en baudios no es tan esencial, ya que los restantes dispositivos son bastante flexibles en las variaciones de velocidad. Se han enviado con éxito caracteres desde el BASIC a través de la RS232 hasta a 2 400 baudios.

Naturalmente, mientras la proporción de bits por byte es de 2 400, el espacio entre bytes es a menudo mucho mayor, de modo que la proporción de transferencia resulta mucho menor.

Pero las cosas cambian cuando se trata de recibir datos. En este caso, incluso a 300 baudios un programa en BASIC apenas tendrá tiempo de captar un byte desde el buffer de entrada y visualizarlo en la pantalla dentro de un bucle simple. Para recibir con efectividad se debe usar algún "control de flujo". Esto significa que se pide al resto de dispositivos que interrumpan su envío hasta que no se llene el buffer. Con el protocolo línea-3 el procedimiento genérico es el siguiente:

1) Leer un pequeño número de bytes (cuanto mayor sea la velocidad de transmisión menor ha de ser este número) empleando GET#2,A$. Tratarlos inmediatamente o colocarlos en una tabla para su posterior proceso.
2) Interrumpir el envío de los otros dispositivos con la instrucción PRINT#2, CHR$(17).
3) Leer más bytes hasta que se vacíe el buffer real de RS232. Procesar cómodamente todos los bytes leídos, mientras se comprueban las teclas pulsadas, señales de final de archivo (EOF), etc.
4) Permitir que el dispositivo envíe de nuevo empleando PRINT#2, CHR$(19) y volver al paso 1.

Cuando se ha abierto un canal RS232, los bytes se envían o reciben de la manera acostumbrada, por medio de PRINT# o GET#. Ha de evitarse el empleo de INPUT#. El byte de estado, ST, debe ser comprobado, lo mismo que exige todo programa con E/S; el estado de error cero se indica por ST=0 o bien ST=8. Por último, CLOSE producirá sin duda otra CLR automática, dado que los buffers son desubicados.

Byte CTRL

Bit 7	6	5	4	3	2	1	0	Función
—	—	—	X	0	0	0	1	50 baudios
—	—	—	X	0	0	1	0	75 baudios
—	—	—	X	0	0	1	1	110 baudios
—	—	—	X	0	1	0	0	134,5 baudios
—	—	—	X	0	1	0	1	150 baudios
—	—	—	X	0	1	1	0	300 baudios
—	—	—	X	0	1	1	1	1 200 baudios
—	—	—	X	1	0	0	0	1 800 baudios
—	—	—	X	1	0	1	0	2 400 baudios
—	0	0	X	—	—	—	—	datos de 8 bits
—	0	1	X	—	—	—	—	datos de 7 bits
—	1	0	X	—	—	—	—	datos de 6 bits
—	1	1	X	—	—	—	—	datos de 5 bits
0	—	—	X	—	—	—	—	1 bit parada
1	—	—	X	—	—	—	—	2 bits parada

X=valor irrelevante

Byte COMM

Bit 7	6	5	4	3	2	1	0	Función
—	—	—	—	X	X	X	0	Protocolo línea-3
—	—	—	—	X	X	X	1	Protocolo línea-X
—	—	—	0	X	X	X	—	*Full duplex*
—	—	—	1	X	X	X	—	*Half duplex*
—	—	0	—	X	X	X	—	Paridad desactivada
0	0	1	—	X	X	X	—	Paridad impar
0	1	1	—	X	X	X	—	Paridad par
1	0	1	—	X	X	X	—	Comprob.-p envío señal desact.
1	1	1	—	X	X	X	—	Comprob.-p envío no señal desact.

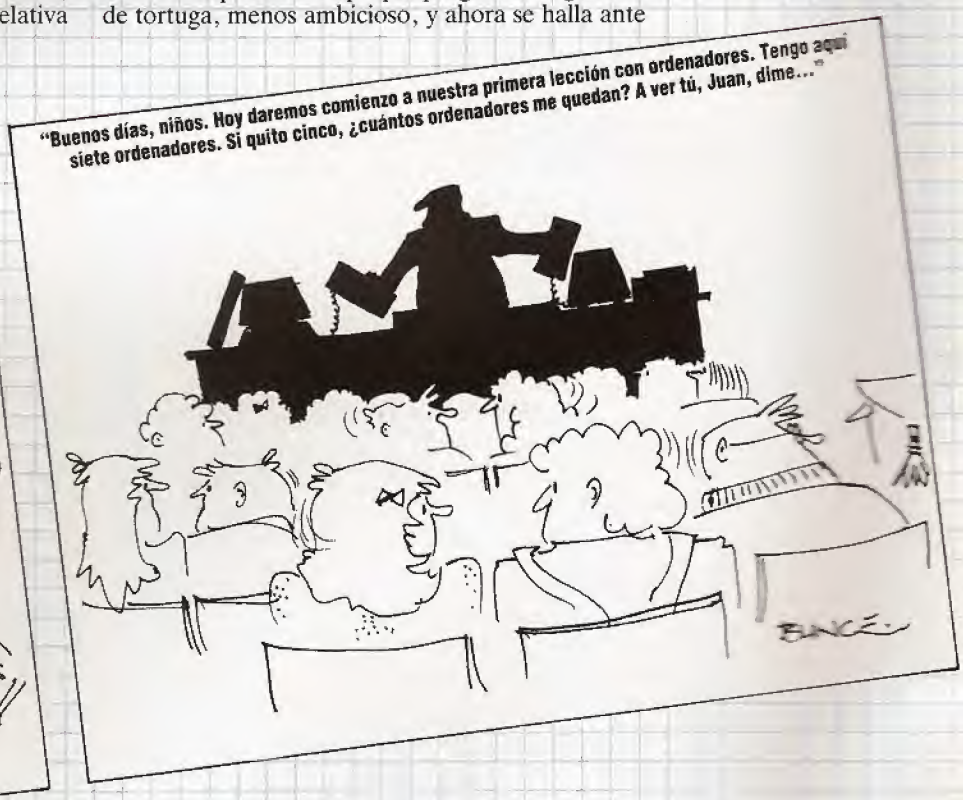
Piedras en el camino

La introducción de la informática en la educación obliga a los maestros a enfrentarse a un sinnúmero de problemas

La introducción de los ordenadores en la educación se está llevando a cabo con algunas dificultades en la mayor parte de los países. Los recortes económicos, la ignorancia generalizada sobre el tema y las dificultades para educar tanto a los futuros maestros como a los maestros en activo, se combinan para imponer severas restricciones en la informática educativa. La nueva tecnología está sufriendo las recientes políticas gubernativas, y los drásticos recortes presupuestarios resultantes para el sector público están creando una escasez de estos recursos en las escuelas. Tras haber adquirido un costoso hardware, muchos maestros están molestos por los presupuestos escolares que no permiten la adquisición de software para poner las máquinas en funcionamiento.

Asimismo, esta falta de financiación ha impedido la puesta en marcha de importantes proyectos de investigación dedicados a la informática y el niño. Se ha prestado muy poca atención a la forma en que influyen los ordenadores en los programas de estudio vigentes y, por consiguiente, no se ha desarrollado una estrategia clara para introducirlos en las escuelas. Tanto el TICCIT y el PLATO, en Estados Unidos, como el Programa Nacional de Desarrollo, en Gran Bretaña, se llevaron a cabo en los años del "pre-micro", a comienzos de los setenta, cuando la financiación podía obtenerse con relativa facilidad. Ahora, en los años ochenta, fiscalmente tan duros, los gobiernos de ambos países no pueden disponer de los recursos para financiar proyectos similares que son urgentemente necesarios. Entre otros problemas, se incluye el hecho de que no se hayan llevado a cabo estudios serios sobre el lugar que ocupan los ordenadores en el plan de estudios. En las escuelas secundarias, la importancia concedida al BASIC y la existencia de la asignatura de informática como disciplina académica por derecho propio están siendo cuestionadas. A las escuelas primarias se les entrega un ordenador y se les dice que "lo hagan servir", y los maestros deben salir del paso a duras penas, de la mejor manera posible.

En ocasiones, dentro del mundo de la informática educativa se producen graves fallos de comunicación. En Londres, muchas escuelas han invertido en máquinas RML y, por consiguiente, el ILECC (Inner London Educational Computing Centre) invirtió muchísimo tiempo y esfuerzos tratando de escribir una versión de LOGO para los micros RML. Mientras tanto, la propia Research Machines Limited producía una implementación completa y estimable del lenguaje. El ILECC, malgastando dinero para intentar recuperar las pérdidas, se dedicó entonces a producir su propio programa de gráficos de tortuga, menos ambicioso, y ahora se halla ante

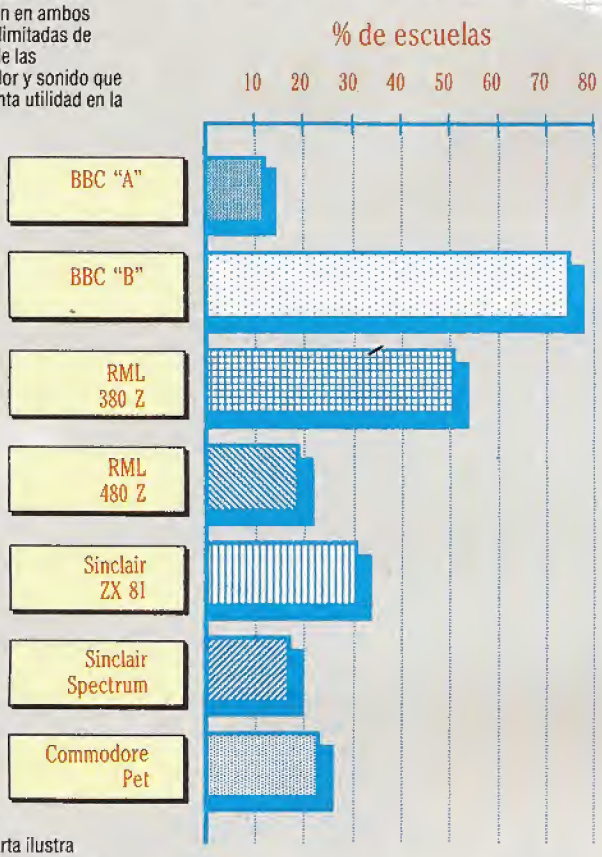


Los favoritos de la clase
Respaldadas por la BBC y recomendadas por el gobierno británico para su uso en las escuelas, las máquinas BBC de Acorn se han asegurado una posición sólida en el sistema educativo del país. Las cifras reflejadas en estas páginas están tomadas de un informe encargado por el gobierno que publicó en enero de 1985 la unidad de investigación de servicios de emisiones educativas de la BBC. Una característica sorprendente de los resultados es la continuada presencia de la antigua serie Pet de Commodore y del Sinclair ZX81, que poseen en ambos casos versiones limitadas de BASIC y carecen de las facilidades de color y sonido que pueden ser de tanta utilidad en la clase

la difícil situación de tener que promocionarlo en las escuelas cuando en el mercado ya existe un producto claramente superior.

Se han planteado otros problemas a consecuencia de la ausencia de una política definida para la formación de los maestros. Se han colocado los ordenadores en las clases, esperando que los maestros aprendieran todo a partir del manual. Las restricciones financieras inhibieron la realización de cursos para maestros en activo y, cuando se dispusieron de los mismos, las escuelas se mostraron reacias a enviar a sus maestros debido a la desorganización que se producía en la escuela debido a su ausencia.

La situación en las escuelas de profesores británicas es aún peor. En 1983, el Departamento de Educación estaba preocupado por la falta de cursos



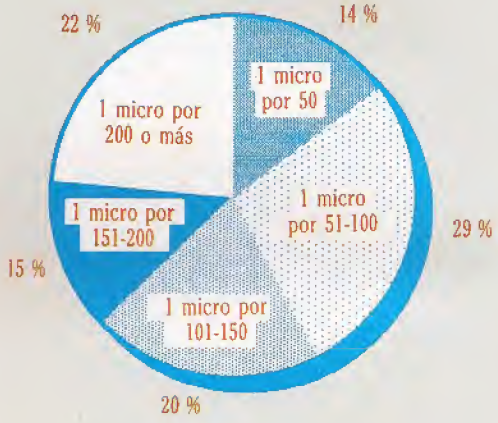
Tarta de micros
Este gráfico de tarta ilustra claramente que a pesar de los intentos del gobierno británico por mejorar la proporción alumnos/micro en las escuelas del país, ¡aún hoy, más de una escuela de cada cinco tiene que arreglarse con menos de un ordenador por cada 200 niños!

Tipos de micros en escuelas

sobre ordenadores para los aspirantes a maestro. Con el apoyo del Departamento de Industria aconsejaron vivamente que cada curso constara de al menos 20 horas dedicadas al papel de los ordenadores en la educación. Chris Gregory, catedrático de Matemáticas en el Bradford College y miembro del MEP (Microcomputers in Education Project, proyecto relacionado con la función del microordenador en la educación), ha afirmado: "Las 20 horas representan un gesto simbólico patético y, aun así, a muchos colegios les resulta difícil implementar siquiera esa cantidad." Un problema acuciante es hallar personas experimentadas. El Departamento de Educación está congelando o recortando el número de personal docente y, en esta situación de reducciones y conflictos con los sindicatos, es difícil reclutar personal con experiencia en ordenadores.

Los programas educativos de calidad han tardado mucho en aparecer. Los gobiernos han invertido en hardware pero no en software y, dado que las escuelas no tienen dinero para gastar, el mercado de software educativo está muy deprimido. En la feria anual de editores de software educativo de 1983 quedó claro que la mayoría de las empresas de software dedicadas a este campo estaban perdiendo dinero y se esperaba que muchas de ellas interrumpieran su actividad durante el año siguiente. Lo que contribuye a deprimir aún más el mercado para los editores de software educativo es el hecho de que muchas de las autoridades más importantes en materia de educación han invertido dinero en desarrollar su propio software, que se distribuye a bajo coste, o gratuitamente, en las escuelas. Esto, como es lógico, hace que los programas comerciales más caros pierdan parte de su atractivo.

Por ejemplo, Sloane Software ha lanzado recientemente un programa de matemáticas denominado *Hopslide*, para niños de entre cinco y siete años. Es un programa sencillo, diseñado para generar una comprensión del valor y el orden de los números. Al niño se le presenta una fila de números entre cinco y nueve. El niño hace saltar los números unos sobre otros o los desplaza una posición. El precio de *Hopslide* es modesto y puede que llegue a venderse bastante bien. La Inner London Education Authority también produce su propio software. El nombre Hopslide también aparece en un programa



Prop. alumnos/micro

de un paquete de matemáticas producido por su centro de matemáticas, SMILE. Se trata del mismo juego que la versión de Sloane Software, con la excepción de que la gama de números es del dos al nueve y que los gráficos son ligeramente mejores. El paquete SMILE contiene otros 19 programas de similar calidad y vale apenas un poco más que el *Hopslide* de Sloane Software.

La falta de financiación y la deficiente organización no se limitan, por supuesto, a la informática educativa; pero existen algunos otros problemas, menos evidentes, que se plantean cuando se intenta introducir la nueva tecnología en las escuelas. La presencia de los ordenadores, por ejemplo, ha causado problemas. En muchas escuelas, equipos elec-

Ian McKinnell

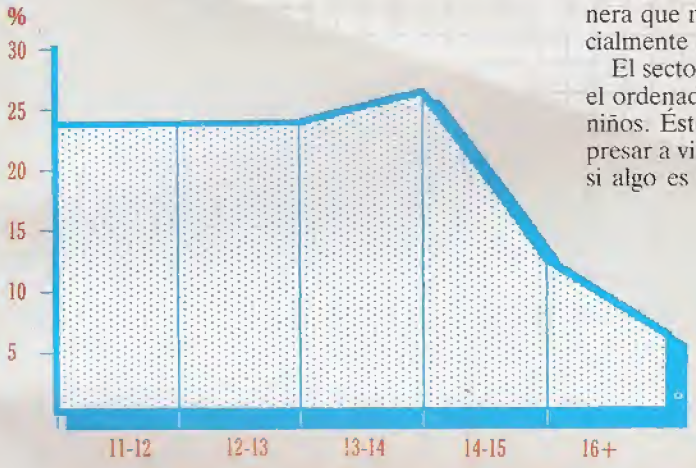
trónicos se almacenan durante la noche en una bóveda de seguridad. Antes de la llegada de los ordenadores, el equipo estaba compuesto por dos o tres cassettes y aparatos de video. En una escuela primaria con un micro por clase, los maestros habían de hacer dos o tres viajes con un ordenador, pantalla, cables, unidad de disco y software a lo largo de un extenso corredor y subir dos plantas por las escaleras, por la mañana y por la tarde. Tras dos accidentes, los maestros se negaron a transportar el equipo y las autoridades educativas correspondientes se negaron a permitir que los ordenadores se dejaran en la clase durante la noche. El resultado fue que todas las máquinas quedaron durante cuatro meses en la bóveda de seguridad, mientras el sindicato de maestros y las autoridades educativas negociaban un acuerdo. Finalmente, el conflicto se resolvió instalando cerca de cada aula armarios de acero para almacenamiento.

En muchas escuelas la situación en materia de seguridad es deplorable, por la sencilla razón de que las autoridades educativas no poseen dinero para invertir en sistemas de seguridad razonables. La mayoría de las escuelas de los barrios centrales superpoblados de las ciudades de Gran Bretaña son objeto de robo o vandalismo varias veces al año. Las pólizas de seguro son tan costosas que a muchas autoridades les resulta más barato reemplazar los bienes sustraídos que asegurarlos.

Entre otros problemas podemos citar el de la discriminación sexual, especialmente porque tanto en la escuela como en el hogar son los varones los que se sienten más atraídos por el ordenador. Los usuarios de la mayoría de los ordenadores personales son varones adolescentes que tienden a utilizarlos para practicar juegos que con frecuencia son violentos y de corte militarista. Esta actitud se ha reproducido en la clase: a menudo el ordenador se considera como un objeto reservado a los varones. Muchos programas educativos están orientados hacia el sexo masculino y, a menudo, los varones monopolizan el ordenador a costa de las mujeres. Un portavoz de la Unidad de Igualdad de Oportunidades de la Inner London Education Authority manifestó: "Somos conscientes de la predilección por el varón en las clases de informática. En las escuelas sólo para mujeres éstas cursan la asignatura con toda satisfacción. El problema sólo se plantea en las escuelas mixtas, donde los niños tienden a dominar. En las escuelas mixtas, pensamos que debemos estimular activamente a las niñas para que se interesen por los ordenadores y restablezcan el equilibrio." Uno de los resultados interesantes que se obtuvieron en la investigación realizada en la Universidad de Edimburgo, utilizando el LOGO en las escuelas, fue que atrajo a las niñas en la misma medida que a los niños.

Otra dificultad importante es que los gobiernos continúan considerando la educación en términos de productividad y costo. Considerar el ordenador como una forma de mejorar la productividad educativa tiene unas implicaciones atemorizadoras para el futuro, con los niños convirtiéndose en parte de una línea de producción informatizada, los maestros reemplazados por máquinas y los planes de estudio y las lecciones determinadas por comisiones del gobierno. Como señaló Seymour Papert: "Existen en el mundo fuerzas formidables que empujan hacia una sociedad más rígida, hacia el empleo del ordenador de una forma que haga la vida menos atractiva, más rígida y menos social. Esas fuerzas son poderosas. ¿Qué podemos hacer frente a ellas?... Podemos decir: 'Quememos los ordenadores.' Eso es una pérdida de tiempo..., o podemos tratar de empujar con la mayor fuerza posible en la dirección contraria y usar el ordenador de una manera que nos permita avanzar en una dirección socialmente deseable."

El sector de la sociedad que finalmente decide si el ordenador es objeto de uso o abuso es el de los niños. Éstos nunca experimentan timidez para expresar a viva voz sus opiniones y son conscientes de si algo es beneficioso y estimulante, o aburrido e irrelevante. Reconocen el software bien pensado y responden a él, y a menudo es tan intensa la atracción que sienten hacia la máquina, que es difícil apartarlos de ésta. En un entorno rico en ordenadores, los niños crean su propia "cultura informática", desarrollando su propia jerga, ayudándose entre sí para resolver los problemas de programación, compartiendo con los demás sus descubrimientos, intercambiando software, educándose a sí mismos colectivamente y evaluando el hardware y el software. Serán los niños, y no los maestros ni los consejeros ni las comisiones ni los investigadores, quienes en última instancia decidan de qué forma se pueden utilizar los ordenadores en la educación para que rindan los mejores frutos.



Experiencia "sobre teclado"

¿Quién falta?
Este gráfico, que representa la experiencia "sobre teclado" de ordenador entre los escolares de Gran Bretaña, muestra los efectos de especialización en el extremo superior de la gama de edades, donde los estudiantes mayores pueden optar por asignaturas que no exijan uso de ordenadores. Nótese que ninguno de los grupos de edades goza de un porcentaje de "experiencia sobre teclado" que supere el 30 %

Buenas relaciones

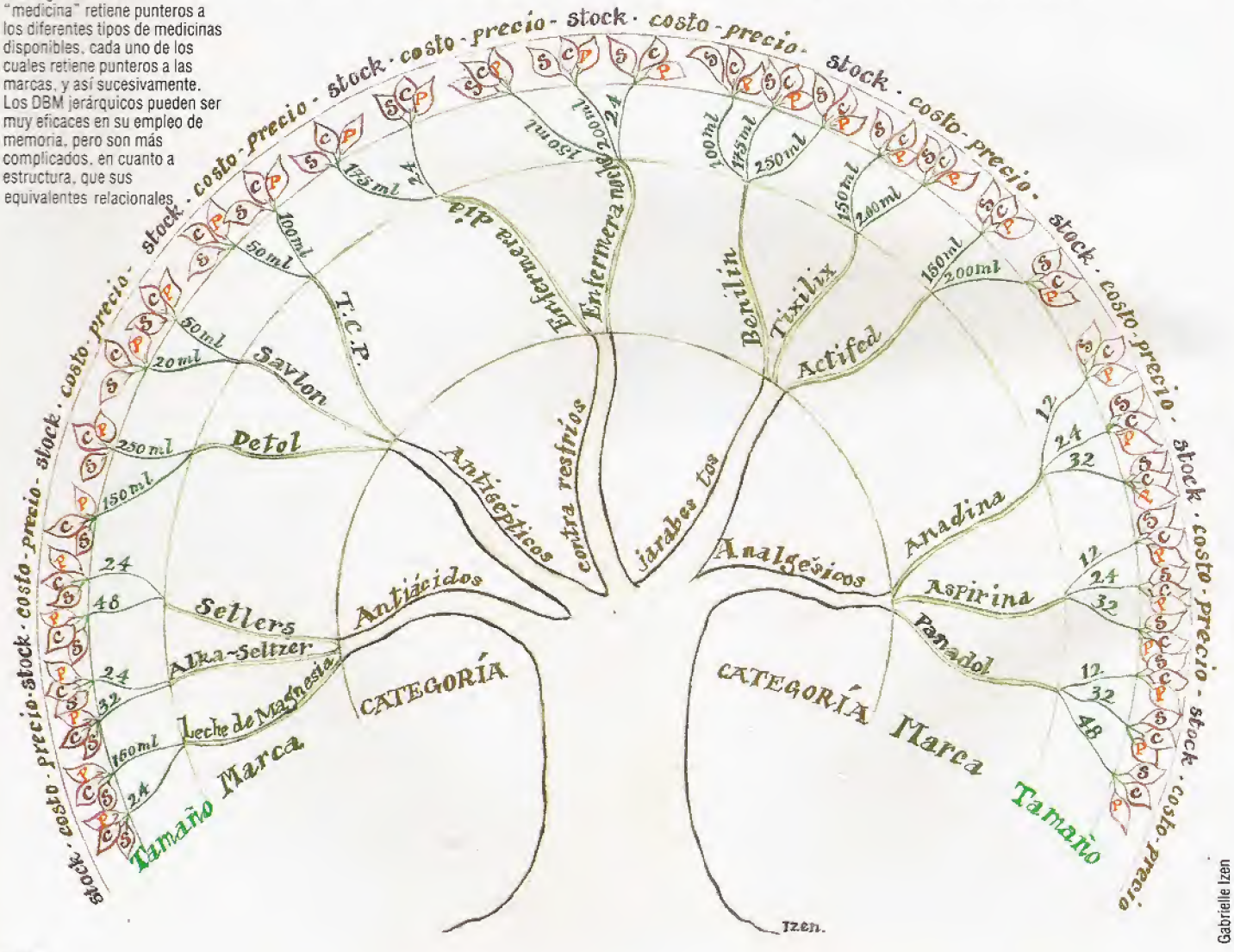
Analizaremos las dos formas más comunes de estructurar una base de datos: los sistemas jerárquicos y en red

La mayor parte de los DBM disponibles en ordenadores personales son del tipo conocido como sistemas *relacionales* de administración de bases de datos. En esencia, esto significa que cada registro de un archivo DBM toma la forma de una tabla, compuesta por filas y columnas, al estilo de una hoja electrónica. Las diversas formas en las que se puede presentar la información sugieren una estructura más compleja, pero ésta, básicamente, no es nada más que una sencilla tabla o cuadrícula. La base de datos se organiza de forma muy parecida a la que emplearía usted para pasar la información en papel. Cada registro es idéntico estructuralmente y cada campo del registro tiene una longitud fija.

El término *relacional* proviene del hecho de que cada "fila" de la base de datos está claramente relacionada, de una forma fija y rígida, a cada "columna". Ésta no es una forma muy flexible de organizar los datos, pero sí sirve para programas administradores de bases de datos relativamente sencillos. Debido a que los ordenadores personales (incluso las nuevas máquinas de 16 bits con 128 K de memoria o más) poseen comparativamente memorias pequeñas, bajas velocidades de proceso y limitadas capacidades para almacenamiento de datos, las restricciones de un sistema relacional forman parte del precio que se debe pagar por combinar una informática al alcance del presupuesto con una administración de bases de datos.

Un enfoque completamente diferente a esta organización tabular de los datos consiste en ordenarlos de forma jerárquica. Podemos pensar que los datos se organizan como si fueran un árbol, cada rama con sus propias bifurcaciones, éstas con sus pequeñas ramitas, e incluso hojas. Para ilustrar esto, crearemos una base de datos del stock de una tienda. En primer lugar, la presentaremos de forma relacional, y después le mostraremos cómo se podría organizar de forma jerárquica.

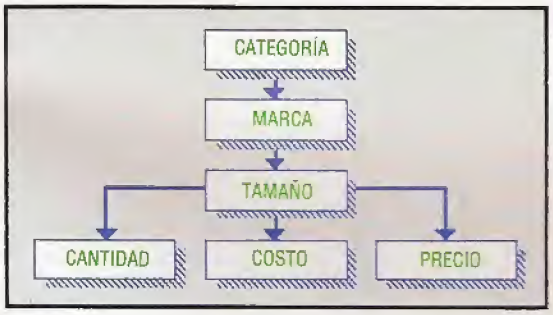
Árbol medicinal
Mientras que un DBM "relacional" almacena la información de forma tabular, un DBM "jerárquico" organiza sus datos en forma de "árbol" En el ejemplo de la ilustración, la categoría individual "medicina" retiene punteros a los diferentes tipos de medicinas disponibles, cada uno de los cuales retiene punteros a las marcas, y así sucesivamente. Los DBM jerárquicos pueden ser muy eficaces en su empleo de memoria, pero son más complicados, en cuanto a estructura, que sus equivalentes relacionales.



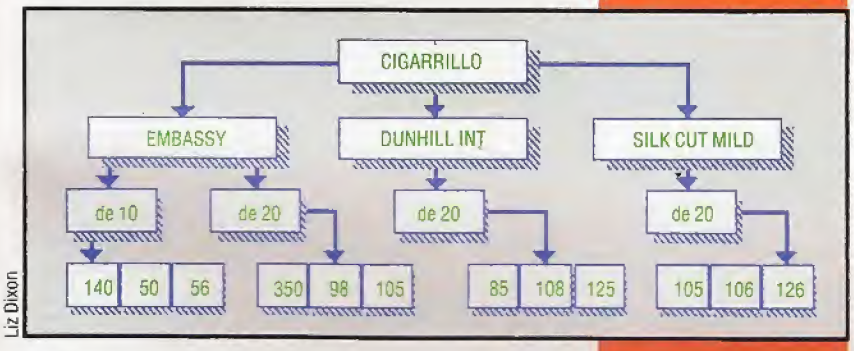
Gabrielle Izen

CATEGORÍA	MARCA	TAMAÑO	STOCK	COSTO	PVP
CIGARRILLO	EMBASSY	20	350	98	105
CIGARRILLO	EMBASSY	10	140	50	56
CIGARRILLO	DUNHILL INT	20	85	108	125
CIGARRILLO	SILK CUT MILD	20	105	106	126
CARAMELOS	BOUNTY	1	106	14	17
CARAMELOS	TWIX	1	95	12	16
CARAMELOS	MARATHON	1	25	15	19
REVISTA	YOUR SPECTRUM	1	35	85	95
REVISTA	NEW STATESMAN	1	12	60	80
REVISTA	CITY LIMITS	1	86	50	60
BEBIDAS	LUCOZADE	150	35	25	37
BEBIDAS	TIZER	150	40	18	27
BEBIDAS	QUOSM	150	20	16	25

Representada jerárquicamente, esta base de datos de stock del quiosco podría tener este aspecto:



El registro, en vez de organizarse por campos, se organiza como "segmentos" de datos, donde cada segmento puede contener más de un campo. Organizadas jerárquicamente en vez de relacionalmente, las entradas correspondientes a CIGARRILLO en la base de datos relacional tendrían este aspecto:



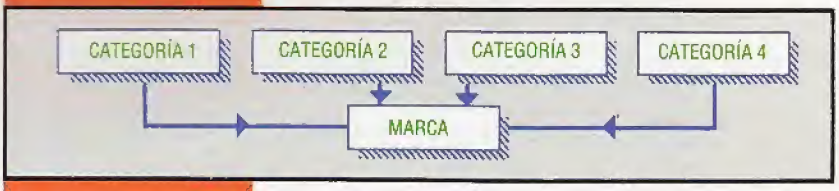
Liz Dixon

Representado de esta forma, cada elemento de la información puede verse claramente como superior o subordinado a otro elemento de ésta. Todo cuanto se requiere es un "puntero" de un segmento de datos a otro. En este ejemplo sólo necesitamos un segmento CIGARRILLO, con punteros a los diversos tipos de cigarrillos que hay en el stock, y otros punteros a tamaño, costo, PVP, etc.

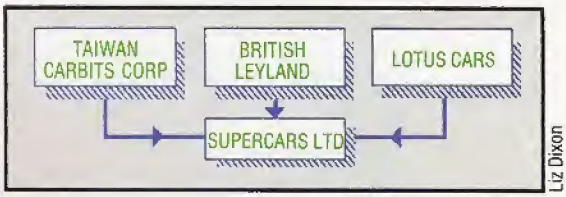
Supongamos que el tendero tiene un centenar de "categorías" en stock, pero mil líneas en venta; con una base de datos relacional, requeriría mil registros: uno para cada línea. Con una base de datos jerárquica, para cubrir todo el stock sólo se necesitaría un centenar de segmentos. Ahorra una gran cantidad de duplicación, pero hace que la base de datos sea más complicada estructuralmente.

Existe aún otro tipo de organización de bases de datos, que se conoce como CODASYL o sistema de red, una organización estandarizada de bases de datos que especificó el grupo de trabajo de bases de datos del Congreso sobre Lenguajes para Sistemas de Datos, CODASYL (Conference on Data Systems Languages). El problema de una base de datos jerárquica es que los datos sólo se pueden organizar para que fluyan en una "dirección"; para utilizar la analogía del árbol, una rama no puede estar unida a dos troncos. Pero, retomando el ejemplo de nuestro tendero, EMBASSY PANATELLAS, por ejemplo, podría ser una "rama" tanto de EMBASSY como de CIGARROS. El problema se plantea particularmente cuando un "componente" simple posee más de un "fabricante". Si usted vende recambios para automóviles, la misma "junta de enlace del extremo mayor" para un BL Marina la podría fabricar la British Leyland, por ejemplo. Como podemos ver, la relación entre mercancías suministradas y proveedores de componentes se puede volver muy complicada.

En el sistema CODASYL se utiliza una red de conjuntos en la que cada conjunto se compone de una colección de registros. La longitud del registro no ha de ser fijada con anterioridad, y cualquier registro puede pertenecer a más de un conjunto. El sistema CODASYL permite que un conjunto esté compuesto por un solo registro, y si hubiera varios conjuntos del mismo tipo, un registro no podría pertenecer a más de uno de ese tipo. Por tanto, esta estructura no está permitida:



Esto representa una limitación seria. Aunque podríamos desear no tener CIGARRILLOs, REVISTAs y BEBIDAs de una misma marca (FABULAN, p. ej.), es fácil imaginar que una empresa llamada Supercars Ltd. formara parte de una estructura como:



Liz Dixon

Si bien las bases de datos jerárquicas y CODASYL son más flexibles que las relacionales, son más complejas y realmente necesitan la potencia de miniordenadores o de ordenadores centrales. Los ordenadores personales utilizan, casi invariablemente, sistemas relacionales. Esta limitación se evidencia en aquellas situaciones en que, por ejemplo, usted tiene un inventario de componentes (libros u otras mercancías) que vende, y cada componente (libro, etc.) posee más de un proveedor. Por lo tanto, necesita un registro de componentes y un conjunto de registros para proveedores relacionados para cada componente. Las DBM de archivos múltiples permiten abrir más de un archivo a la vez y que el archivo activo se remita a un archivo subsidiario.

Puertas abiertas

Veamos cómo trabajan los dos chips CIA 6526 que controlan la comunicación con el exterior en el Commodore 64, analizando el programa "Parawedge"

El Commodore 64 tiene dos chips 6526 CIA (Adaptador de interface complejo), encargados de la comunicación con el mundo exterior. No son ellos los únicos chips con capacidades de E/S; tanto el chip 6510 como el de video manejan también aspectos de E/S. Un chip 6526 tiene dos puertas de datos de ocho bits, ambas con líneas programables por separado. El chip soporta la comunicación de 16 bits o de 8 bits y tiene dos temporizadores de 16 bits enlazables. Posee además un registro de desplazamiento de 8 bits para la comunicación en serie, y, como ya vimos, un reloj de hora-del-día de 24 horas programable.

Tiene además dos líneas específicas de enlace (*handshaking*): la PC y la Flag. La PC bajará durante un ciclo después de que los datos sean escritos en la puerta B del 6526, y puede emplearse para indicar a un dispositivo externo que están preparados los datos (*data ready*). La línea Flag puede usarse como una entrada de control desde otro dispositivo.

Esto puede servir para poner a uno el bit Flag en el registro de interrupción y, si es el caso, originar una interrupción no enmascarable (NMI) al procesador 6510.

En el Commodore 64 los dos chips 6526 tratan aspectos diversos de las E/S: uno de ellos (CIA#1, con la dirección de base en $DC00) se destina al teclado y a las palancas de mando, mientras que el otro (CIA#2, con dirección de base en $DD00) controla los datos en las puertas seriales y del usuario. El chip video maneja las E/S al monitor, y la puerta de cassette es tratada directamente por el 6510.

El programa Parawedge que damos aquí ilustra con claridad los pasos necesarios para programar el 6526 directamente para E/S. La rutina es una cuña NMI que emplea la línea Flag. Está diseñada para enviar un bloque específico de la memoria por la puerta para el usuario como datos paralelos, o bien para recibir datos paralelos de 8 bits hasta que el bloque especificado de la memoria esté lleno. Dado que se trata de una codificación de cuña, temporizará los datos que entran o salen sobre NMIs, dejando la máquina libre para que realice otras tareas en el tiempo restante. El único inconveniente es que si la velocidad de transmisión de datos es demasiado alta el Commodore 64 comenzará a gastar todo el tiempo ejecutando rutinas de servicio NMI, lo que puede ser muy engorroso.

El programa Parawedge permite que el Commodore 64 establezca comunicaciones a dos vías de 8 bits en paralelo con un dispositivo externo (otro ordenador, una impresora en paralelo) a través de la puerta para el usuario. Las patillas de la puerta para el usuario desde PB0 hasta PB7 se emplean para el traslado de datos; la patilla Flag 2 se emplea como entrada de enlace; la PA2 da la señal de *ready for data* (preparado para datos), y la PC2 da la señal *data valid* (dato válido). Para emplear el programa, se ha de definir primero un área de RAM desde la cual se desea enviar los datos que salen, o sobre la cual se desea recibir los datos entrantes. Esto se consigue pasando al programa las direcciones de inicio y de final (en la forma *lo-hi*) colocándolas (POKE) en las cuatro posiciones que van de la 50768 a la 50771.

La posición 50772 se emplea para indicar si el programa va a realizar una operación de Entrada o Salida de datos. Si se coloca (POKE) un uno en esta posición el programa quedará preparado para salida; si se coloca un cero, el programa estará en modo entrada. Una vez establecidos estos parámetros, la codificación de cuña se inicia por medio de SYS 50775. Dado que el programa está conducido por interrupciones, se ejecutará entre bastidores, enviando o recibiendo datos, mientras se digita o ejecuta un programa cualquiera en BASIC en primer plano.

Programa Parawedge para el Commodore 64

Cargador del BASIC

```
1000 REM ** CARGADOR DEL BASIC DEL PARAWEDGE **
1010 DATA173,84,198,208,61,169,0,141,3
1020 DATA221,169,144,141,13,221,173,2
1030 DATA221,9,4,141,2,221,173,0,221,9
1040 DATA4,141,0,221,173,80,198,133,251
1050 DATA173,81,198,133,252,173,24,3
1060 DATA141,85,198,173,25,3,141,86,198
1070 DATA120,169,188,141,24,3,169,198
1080 DATA141,25,3,88,96,169,255,141,3
1090 DATA221,169,144,141,13,221,173,24
1100 DATA3,141,85,198,173,25,3,141,86
1110 DATA198,120,169,234,141,24,3,169
1120 DATA198,141,25,3,88,96,169,144,44
1130 DATA13,221,240,36,173,1,221,145
1140 DATA251,230,251,208,2,230,252,173
1150 DATA82,198,197,251,173,83,198,229
1160 DATA252,144,49,173,0,221,41,252
1170 DATA141,0,221,9,4,141,0,221,108,85
1180 DATA198,169,144,44,13,221,240,246
1190 DATA177,251,141,1,221,230,251,208
1200 DATA2,230,252,173,82,198,197,251
1210 DATA173,83,198,229,252,144,3,108
1220 DATA85,198,120,173,85,198,141,24,3
1230 DATA173,86,198,141,25,3,88,108,24
1240 DATA3
1250 DATA25596:REM"SUMA DE CONTROL"
1260 CC=0
1270 FOR I=50775TO50971
1280 READX:CC=CC+X:POKE I,X
1290 NEXT
1300 READX:IFCC<>XTHENPRINT"ERROR SUMA DE
     CONTROL"
1310 END
```

Entrada del Parawedge
Incluimos aquí el listado en código assembly del Parawedge, que puede digitarse y ensamblarse por medio de un ensamblador. El programa puede, igualmente, ser introducido como una serie de sentencias DATA, digitando y ejecutando la codificación del cargador del BASIC

Listado en assembly

```
;*************************************************
;*                                               *
;*  PARAWEDGE - PROGRAMA CUÑA DE ENVIO/RECEPCION *
;*                                               *
;*          EN COMUNICACIONES DE 8 BITS          *
;*                                               *
;*          EN PARALELO EN EL COMMODORE 64       *
;*                                               *
;*************************************************

CIA 2=$DD00          ; DIRECCION DE BASE DEL CHIP 6526
OUTPUT=$FF
INPUT=$00
OUTSHK=$04
INTMSK=$90
TOGHI=$04
TOGLO=$FC
NMIVEC=$0318
ZPTEMP=$FB
*=$C650
START *=*+2          ; DIRECCION DE INICIO
END *=*+2            ; DIRECCION DEL FINAL
MODE *=*+1           ; FLAG DE E/S
VECTOR *=*+2         ; ALMACEN PARA EL VECTOR NMI

  LDA  MODE          ;ENTRADA O SALIDA
  BNE  OUTDAT        ;BIFURCACION SI ES SALIDA
  LDA  #INPUT
  STA  CIA2+3        ;ESTABLECE DDR PARA ENTRADA
  LDA  # INTMSK
  STA  CIA2+13       ;DESACTIVA LAS INTERRUPCIONES DEL FLAG
  LDA  CIA2+2
  ORA  #OUTSHK
  STA  CIA2+2        ;PONE PA2 PARA SALIDA
  LDA  CIA2
  ORA  #TOGHI
  STA  CIA2          ;PONE ALTA LA LINEA PA2 DE ENLACE
  LDA  START
  STA  ZPTEMP
  LDA  START+1       ;MUEVE PUNTEROS A PAGINA CERO
  STA  ZPTEMP+1
;
;INICIALIZACION CUÑA ENTRADA
;
  LDA  NMIVEC
  STA  VECTOR        ;GUARDA EL ANTIGUO VECTOR NMI
  LDA  NMIVEC+1
  STA  VECTOR+1
  SEI                ;
  LDA  #<NXTIN       ;
  STA  NMIVEC        ;INSERTA LA CUÑA DE ENTRADA DATOS
  LDA  #>NXTIN       ;
  STA  NMIVEC+1      ;
  CLI                ;
  RTS
;
;INICIALIZACION CUÑA SALIDA
;
OUTDAT
  LDA  #OUTPUT
  STA  CIA2+3        ;PONE DDR PARA SALIDA
  LDA  #INTMSK
  STA  CIA2+13       ;DESACTIVA INTERRUPCIONES FLAG
;
;
  LDA  NMIVEC
  STA  VECTOR
  LDA  NMIVEC+1      ;GUARDA ANTIGUO VECTOR NMI
  STA  VECTOR+1
  SEI                ;
  LDA  #<NXTOUT      ;
  STA  NMIVEC        ;INSERTA LA CUÑA DE SALIDA DATOS
  LDA  #>NXTOUT      ;
  STA  NMIVEC+1      ;
  CLI                ;
  RTS
;
;
;RUTINA DE SERVICIO DATOS ENTRADA
;
;
NXTIN
  LDA  #INTMSK       ;COMPRUEBA ICR
  BIT  CIA2+13       ;INTERR DEBIDA A FLAG?
  BEQ  NOTCOM        ;NO... NMI NORMAL
;
;BYTE OK EN PUERTA
;
  LDA  CIA2+1        ;LEE EL BYTE
  STA  (ZPTEMP),Y    ;LO ALMACENA EN MEMORIA
  INC  ZPTEMP
  BNE  TEST1         ;INCREMENTA EL PUNTERO
  INC  ZPTEMP+1
;
TEST1
  LDA  END
  CMP  ZPTEMP
  LDA  END+1         ;COMPRUEBA SI ES EL FINAL
  SBC  ZPTEMP+1
  BCC  DONE          ;BIFURCACION SI SE HA ACABADO
;
;AVISA A DISPOSITIVO PREPARADO PARA BYTE SIGUIENTE
;
  LDA  CIA2
  AND  #TOGLO
  STA  CIA2          ;PONE PA2 BAJA Y DESPUES ALTA
  ORA  #TOGHI
  STA  CIA2
;
;REALIZA AHORA LA RUTINA NORMAL NMI
;
NOTCOM
  JMP (VECTOR)


;
;RUTINA DE SERVICIO DATOS SALIDA
;
NXTOUT
  LDA  #INTMSK       ;COMPRUEBA ICR
  BIT  CIA2+13       ;INTERR. DEBIDA AL FLAG?
  BEQ  NOTCOM        ;NO... EFECTUA NMI NORMAL
;
;ENVIO OK BYTE
;
  LDA  (ZPTEMP),Y    ;TOMA BYTE DE LA MEMORIA
  STA  CIA2+1        ;LO SACA.EL PC BAJARA
                     ;DURANTE UN CICLO
  INC  ZPTEMP
  BNE  TEST2         ;INCREMENTA EL PUNTERO
  INC  ZPTEMP+1

TEST2
  LDA  END
  CMP  ZPTEMP
  LDA  END+1         ;COMPRUEBA SI ES EL FINAL
  SBC  ZPTEMP+1
  BCC  DONE          ;BIFURCACION SI SE HA ACABADO
;
;CONTINUA RUTINA NMI NORMAL
;
  JMP (VECTOR)
;
;TERMINADO QUITAR CUÑA
;
DONE
  SEI
  LDA  VECTOR
  STA  NMIVEC        ;RESTAURA EL VECTOR NMI A SU
  LDA  VECTOR+1      ;VALOR INICIAL
  STA  NMIVEC+1
  CLI
  JMP  (NMIVEC)
```

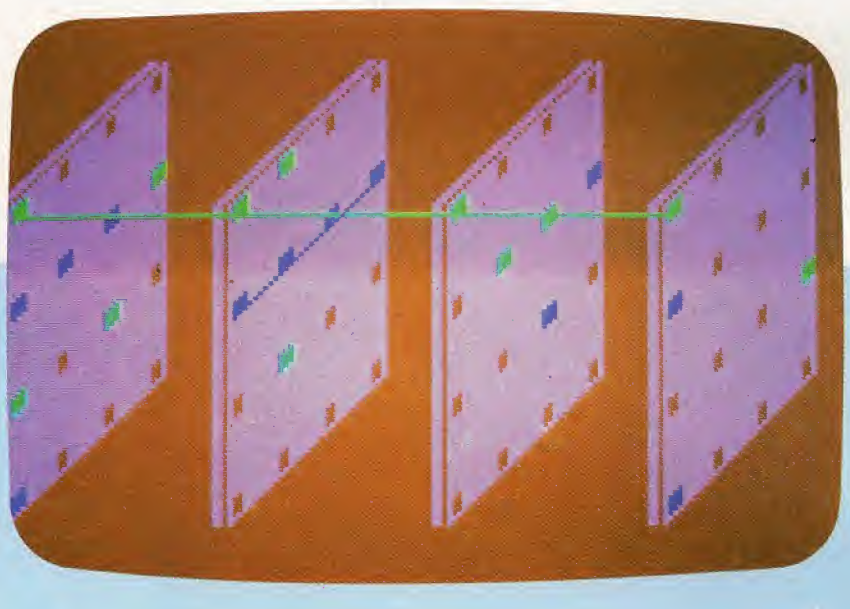
Parawedge está tomado de *Mastering the Commodore*, de Jones y Carpenter, y se imprime por cortesía de los autores y de la Ellis Horwood Ltd.

Cuatro en raya

Este juego en tres dimensiones exige 24 K de memoria. Le permite jugar contra el ordenador Atari u otro adversario de su elección

Para ganar hay que colocar cuatro cuadrados de igual color en línea. Usted juega en 4 planos (A, B, C, D), divididos en 16 cuadrados. Para jugar, primero debe elegir un plano y luego digitar un número correspondiente al cuadrado escogido. Éstos están numerados así:

13	14	15	16
09	10	11	12
05	06	07	08
01	02	03	04



He aquí ejemplos de pulsaciones (recuerde que es inútil pulsar la tecla "Return"): D15, A05, B10, B16, C02. Cuando el ordenador le solicite el número de jugadores, digite 1 para jugar contra la máquina o 2 para enfrentar a otro adversario.

```
 0 REM ******************************
 1 REM *          CUATRO EN RAYA      *
 2 REM *      EN TRES DIMENSIONES     *
 3 REM ******************************
 4 GOSUB 3000
 5 DIM SPACE(64,2):DIM MARK(64):DIM BLOB(
   86,3):DIM HZ(40)
 6 CL=0
 7 GOSUB 6000
10 GRAPHICS 7+16:COLOR 1
17 SETCOLOR 2,12,8:SETCOLOR 4,0,3
18 SETCOLOR 1,4,8:SETCOLOR 0,2,6:SETCOLO
   R 3,3,8
20 FOR X=0 TO 129 STEP 43
30 FOR Y=95 TO 30 STEP -1
40 PLOT X,Y:DRAWTO X+30,Y-30
50 NEXT Y
60 NEXT X
61 PLOT 0,28:DRAWTO 28,0:PLOT 0,27:DRAWT
   O 27,0
62 FOR RT=0 TO 86 STEP 43
63 PLOT RT+41,95:DRAWTO RT+41,30:DRAWTO
   RT+71,0:PLOT RT+40,95:DRAWTO RT+40,30:DR
   AWTO RT+70,0
64 NEXT RT
65 COLOR 0:SETCOLOR 4,9,2
66 FOR Z=0 TO 129 STEP 43
70 FOR Y=91 TO 31 STEP -20
72 FOR X=1 TO 31 STEP 9
74 FOR A=0 TO 1
760 PLOT Z+X+A,Y-A-X+3:DRAWTO Z+X+A,Y-A-X
780 NEXT A
850 NEXT X
900 NEXT Y
950 NEXT Z
1000 FOR P=1 TO 64:READ H,J
1010 SPACE(P,1)=H:SPACE(P,2)=J
1012 NEXT P
1015 IF PEEK(764)=255 THEN 1015
1016 CLOSE #1:OPEN #1,4,0,"K:":GET #1,H
1017 IF W<65 OR H>68 THEN POKE 764,255:6
     OTO 1015
1018 IF W=65 THEN N=0:IF W=66 THEN N=16:
     IF W=67 THEN N=32
1019 IF W=66 THEN N=16
1020 IF W=67 THEN N=32
1021 IF W=68 THEN N=48
1023 POKE 764,255
1024 IF PEEK(764)=255 THEN 1024
1025 CLOSE #5:OPEN #5,4,0,"K:":GET #5,Z
1026 IF Z<48 OR Z>49 THEN POKE 764,255:6
     OTO 1024
1027 POKE 764,255
1028 POKE #4:OPEN #4,4,0,"K:":GET #4,HH
     :IF WH<48 OR WH>57 THEN POKE 764,255:GOT
     O 1027
1033 REM FOR TR=0 TO 2
1034 SP=N+10*(Z-48)+WH-48:LOCATE
     SPACE(SP,1),SPACE(SP,2),GH:IF GH<>0 THEN
     GOTO 1015
1035 CL=CL+1:IF CL=2 THEN CL=0
1036 FOR RY=15 TO 0 STEP -1:COLOR (RY+CL
     +2):SETCOLOR 1,3,4
1037 FOR TR=0 TO 2
1038 PLOT SPACE(SP,1)+TR,SPACE(SP,2)-TR:
     DRAWTO SPACE(SP,1)+TR,SPACE(SP,2)-TR-3
1039 NEXT TR:NEXT RY
1040 CLOSE #1
1042 FOR PQ=1 TO 64
1044 LOCATE SPACE(PQ,1),SPACE(PQ,2),V
1046 MARK(PQ)=V:NEXT PQ
1050 C=CL+2
1100 A=1:B=16:D=16:E=1:GOSUB 1400
1105 A=1:B=61:D=1:E=4:GOSUB 1400
1110 A=1:B=13:D=17:E=4:GOSUB 1400
1115 A=4:B=16:D=15:E=4:GOSUB 1400
1120 A=1:B=4:D=20:E=1:GOSUB 1400
1125 A=13:B=16:D=12:E=1:GOSUB 1400
1130 A=1:B=1:D=21:E=1:GOSUB 1400
1135 A=13:B=13:D=13:E=1:GOSUB 1400
1140 A=16:B=16:D=11:E=1:GOSUB 1400
1145 A=4:B=4:D=19:E=1:GOSUB 1400
1150 A=1:B=49:D=5:E=16:GOSUB 1400
1155 A=4:B=52:D=3:E=16:GOSUB 1400
1160 A=1:B=4:D=4:E=1:GOSUB 1400
1165 A=17:B=20:D=4:E=1:GOSUB 1400
1170 A=33:B=36:D=4:E=1:GOSUB 1400
1175 A=33:B=36:D=4:E=1:GOSUB 1400
1180 A=49:B=52:D=4:E=1:GOSUB 1400
1299 IF JOUEUR=1 AND CL<>0 THEN GOTO 150
     0:REM IF CL=0 THEN GOTO 1015
1300 GOTO 1015
1400 FOR PQ=A TO B STEP E
1410 IF MARK(PQ)=C AND MARK(PQ+D)=C AND
     MARK(PQ+2*D)=C AND MARK(PQ+3*D)=C THEN G
     OSUB 1450
1420 NEXT PQ
1430 RETURN
1450 PLOT SPACE(PQ,1)+1,SPACE(PQ,2)-1:DR
     AWTO SPACE(PQ+3*D,1)+1,SPACE(PQ+3*D,2)-1
     :RETURN
1500 REM JUEGO ORDENADOR
1600 FOR PQ=1 TO 64
1605 LOCATE SPACE(PQ,1),SPACE(PQ,2),V
1606 IF V=2 THEN V=10
1608 IF V=3 THEN V=50
1610 MARK(PQ)=V:NEXT PQ
1615 H=0
1620 A=1:B=16:D=16:E=1:GOSUB 1700
1625 A=1:B=61:D=1:E=4:GOSUB 1700
1630 A=1:B=13:D=17:E=4:GOSUB 1700
1635 A=4:B=16:D=15:E=4:GOSUB 1700
1640 A=1:B=4:D=20:E=1:GOSUB 1700
1645 A=13:B=16:D=12:E=1:GOSUB 1700
1650 A=1:B=1:D=21:E=1:GOSUB 1700
1655 A=13:B=13:D=13:E=1:GOSUB 1700
1660 A=16:B=16:D=11:E=1:GOSUB 1700
1665 A=4:B=4:D=19:E=1:GOSUB 1700
1670 A=1:B=49:D=5:E=16:GOSUB 1700
1675 A=4:B=52:D=3:E=16:GOSUB 1700
1680 A=1:B=4:D=4:E=1:GOSUB 1700
1682 A=17:B=20:D=4:E=1:GOSUB 1700
1686 A=33:B=36:D=4:E=1:GOSUB 1700
1688 A=49:B=52:D=4:E=1:GOSUB 1700
1690 GOTO 1740
1700 FOR PQ=A TO B STEP E
1705 H=H+1
1710 SUM=MARK(PQ)+MARK(PQ+D)+MARK(PQ+2*D
     )+MARK(PQ+3*D)
1720 IF SUM=30 THEN GOTO 1830
1725 BLOB(H,1)=SUM:BLOB(H,2)=BLOB(H,3)=D
1730 NEXT PQ:RETURN
1740 FOR TZ=1 TO H:IF BLOB(TZ,1)=150 THE
     N GOTO 1850
1745 NEXT TZ
1746 YZ=0:FOR TZ=1 TO H
1748 IF BLOB(TZ,1)=20 OR BLOB(TZ,1)=100
     THEN GOSUB 1860
1750 NEXT TZ:IF YZ<>0 THEN GOTO 1760
1755 GOTO 2050
1760 FOR TZ=1 TO YZ:FOR PZ=2 TO (YZ-1)
1765 IF HZ(TZ)=HZ(PZ) AND TZ<>PZ THEN SP
     =HZ(TZ):GOTO 1035
1766 NEXT PZ:NEXT TZ
1770 IF YZ<>0 THEN SP=HZ(YZ):GOTO 1035
1825 GOTO 2050
1830 FOR TK=0 TO 3:IF MARK(PQ+TK*D)=0 TH
     EN SP=PQ+TK*D:GOTO 1035
1835 NEXT TK
1850 REM DEFENSA
1855 FOR TK=0 TO 3:IF MARK(BLOB(TZ,2)+TK
     *BLOB(TZ,3)=0 THEN SP=BLOB(TZ,2)+TK*BLO
     B(TZ,3):GOTO 1035
1857 NET TK
1860 FOR P=0 TO 3
1863 IF MARK(BLOB(TZ,2)+P*BOB(TZ,3))=0
     THEN YZ=YZ+1:HZ(YZ)=BLOB(TZ,2)+P*BLOB(TZ
     ,3)
1865 NEXT P:RETURN
1890 GOTO 2050
1950 GOTO 2050
2050 SP=INT(64*RND(1)+1):IF MARK(SP)=0 T
     HEN GOTO 1035
2055 GOTO 2050
3000 REM PANTALLA ENTERA
3020 GRAPHICS 2+16:SETCOLOR 4,13,2
3030 POSITION 3,1:?#6;" ################ "
3040 POSITION 3,2:?#6;"#          CUATRO      #"
3050 POSITION 3,3:?#6;"#          EN RAYA     #"
3052 POSITION 3,4:?#6;"#          EN TRES     #"
3054 POSITION 3,5:?#6;"#        DIMENSIONES   #"
3060 POSITION 3,6:?#6;" ################ "
3070 POSITION 1,7:?#6;"       PULSE START    "
3080 IF PEEK(53279)=6 THEN RETURN
3090 GOTO 3080
5000 DATA 1,93,10,84,19,75,28,66
5010 DATA 1,73,10,64,19,55,28,46
5020 DATA 1,53,10,44,19,35,28,26
5030 DATA 1,33,10,24,19,15,28,6
5040 DATA 44,93,53,84,62,75,71,66
5050 DATA 44,73,53,64,62,55,71,46
5060 DATA 44,53,53,44,62,35,71,26
5070 DATA 44,33,53,24,62,15,71,6
5080 DATA 87,93,96,84,105,75,114,66
5090 DATA 87,73,96,64,105,55,114,46
5100 DATA 87,53,96,44,105,35,114,26
5110 DATA 87,33,96,24,105,15,114,6
5120 DATA 130,93,139,84,148,75,157,66
5130 DATA 130,73,139,64,148,55,157,46
5140 DATA 130,53,139,44,148,35,157,26
5150 DATA 130,33,139,24,148,15,157,6
6000 GRAPHICS 2+16:? #6;" 2 jugadores 0"
6010 ? #6;"1 JUGADOR CONTRA MI"
6020 ? #6;"(PULSE 1 O 2)"
6030 OPEN #1,4,0,"K:":GET #1,M:IF M<49 O
     R M>50 THEN POKE 764,255:CLOSE #1:GOTO 6
     030
6040 JOUEUR=M-48:CLOSE #1
6050 RETURN
```

Tres en uno

El Penman Plotter, creado para el BBC Micro, es un dispositivo que puede funcionar como trazador de gráficos, tortuga o "ratón"

La gran cantidad de interfaces existentes en el BBC Micro, así como su amplia aceptación entre los especialistas de la educación, se han combinado para producir una gran cantidad de periféricos de carácter didáctico para la máquina. Aquí ya hemos examinado un gran número de dispositivos que se pueden controlar mediante el BBC Micro, incluyendo máquinas tan diversas como plotters (dispositivos trazadores de gráficos) ratones y robots móviles. No obstante, el Penman Plotter es un periférico que con toda justicia se puede considerar como estos tres dispositivos fusionados en uno.

El paquete Penman Plotter se compone de una unidad de control, un plotter móvil, una fuente de alimentación eléctrica, un cable conector RS232 y el software que lo acompaña. Cuando no se está utilizando, la unidad de control y el plotter se unen entre sí para conformar una única unidad compacta, que mide sólo 55 por 128 por 335 mm. Para separar el plotter de la unidad de control sólo hay que pulsar el clip que hay debajo de la misma, deslizar el plotter hacia afuera y desenrollar el cable plano conector desde el interior mismo de la unidad de control. Una vez extraído de su lugar, el dispositivo deja al descubierto tres agujeros en los que se colocan los lápices, así como una copa situada exactamente en el medio, donde se puede colocar otro lápiz para gráficos de tipo tortuga. Los lápices, de punta de fieltro y 40 mm de longitud, una vez colocados en sus copas, se mantienen sobre el papel mediante clips sustentados por resortes. Cuando se da la orden de dibujar, el resorte es empujado hacia abajo por los electroimanes situados en las barras de dentro, y el peso del lápiz presiona la punta contra el papel.

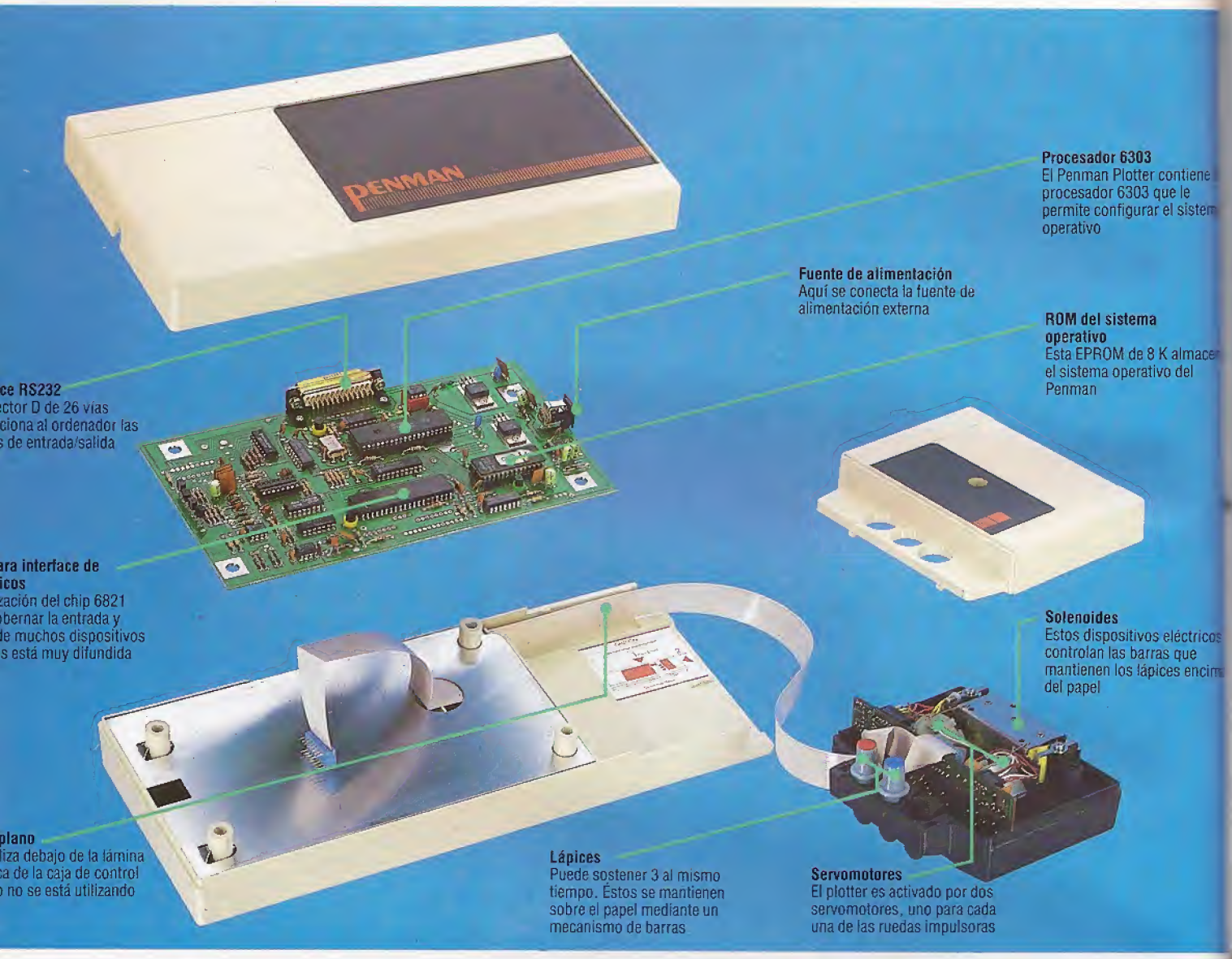
En la cara inferior del dispositivo hay tres ruedas, dos de las cuales son propulsoras; la tercera es una pequeña rueda plástica de movimiento libre que se encarga del equilibrio. Las ruedas propulsoras son de metal y están rodeadas por una cubierta gruesa para proporcionar una fricción adicional cuando el plotter se desplaza sobre el papel. En frente de cada una de ellas hay un sensor luminoso, que detecta el margen del papel registrando la diferencia de brillo entre el papel y el material sobre el cual está colocado.

Dentro del dispositivo trazador de gráficos hay tres electroimanes conectados a barras, cada uno de los cuales controla la elevación y el descenso de un lápiz sobre el papel; junto a ellos se halla el par de motores eléctricos estándares unidos a las ruedas impulsoras. El plotter utiliza motores eléctricos comunes en lugar de motores paso a paso o servo; éstos permiten que el dispositivo trace curvas continuas en vez de curvas "paso a paso". Sin embargo, ello exige que el software de control sea paralelamente más sofisticado, porque ha de ser capaz de variar los voltajes aplicados a los motores, en vez de limitarse a enviar una serie de impulsos.

Por último, hay una placa de circuitos que distribuye la potencia y decodifica las señales que llegan desde la unidad de control y que las envía ya sea a los electroimanes o a los motores. En cada uno de los husillos de los motores hay un disco estroboscópico que gira a medida que rota el motor, y a los lados de las caras del disco hay diodos detectores de luz. A medida que el disco estroboscópico gira, en los diodos aparecen impulsos de luz que le informan a la lógica de la placa de circuitos sobre la rapi-

Chris Stevens

Doble personalidad
El Penman Plotter se compone de un módulo de control, conectado a un ordenador a través de una interface RS232, y una tortuga móvil. El cable plano que une la tortuga a la caja de control es bidireccional, lo que significa que los dos dispositivos se pueden comunicar entre sí. Por consiguiente, mientras la caja de control le envía instrucciones a la tortuga para que se desplace a una posición, la tortuga puede transmitir su posición de vuelta al sistema de circuitos de control. Es esta característica del sistema la que permite utilizar el Penman Plotter no sólo como tortuga y plotter, sino tambien como "ratón"



dez a que se está moviendo el motor, de modo que la máquina pueda calcular su posición. Esta lógica es particularmente útil cuando se utiliza el Penman como "ratón", porque el ordenador necesita saber en qué dirección y a qué velocidad están girando las ruedas para poder desplazar el cursor a través de la pantalla.

La placa de circuito impreso del interior de la unidad de control tiene instalados tres chips principales. La placa contiene un microprocesador 6303 (un desarrollo del 6800), que posee la facilidad de configurar toda una variedad de sistemas operativos. El procesador incorpora una pequeña RAM de trabajo que le permite almacenar la posición del plotter. El segundo de los chips principales de la placa es una ROM para interface de periféricos 6821. También incluida en la placa hay una EPROM que contiene los programas de demostración que llevará a cabo el Penman cuando no esté conectado el cable RS232.

El Penman se puede utilizar en varias modalidades diferentes. La modalidad "emulador de terminal" permite controlar el dispositivo directamente desde el teclado. Para entrar en ella, se debe cargar directamente el programa *Pentlk* o, empleando el robot Penman como ratón, cargar el programa desde el menú principal. Una vez cargado el software, las instrucciones se envían a la unidad de control a través de la puerta RS232 en forma de códigos ASCII. Digitando PRINT 'I' se inicializa el Penman. Él recibirá la señal y determinará cuál de las tres velocidades posibles (300, 1 200, 9 600 baudios) se está utilizando. Hecho esto, el Penman se puede colocar en su secuencia de colocación en la posición inicial digitando la instrucción H. El robot realizará una secuencia de movimientos que hará que el dispositivo se sitúe en el rincón inferior izquierdo del papel.

Desde su posición de partida, el Penman intentará hallar la parte inferior de la página mediante el empleo de sus sensores detectores de luz. Una vez hallado el margen, el dispositivo realizará un giro de 90° y llevará a cabo la misma acción a lo largo del lado izquierdo de la página. Como ayuda para asegurar que el contraste entre el papel y el fondo sea suficiente para que el plotter lo pueda detectar, Penman Products ha incluido sabiamente un trozo de papel negro sobre el cual se puede colocar el papel de dibujo. Una vez que el Penman ha "aterrizado" en su base, establece su posición de partida a

50 mm de cada borde del papel (la distancia desde la parte frontal del plotter hasta la copa central).

La aplicación *Pentlk* se puede ejecutar ya sea en modalidad directa (una instrucción cada vez) o bien se pueden encadenar instrucciones entre sí para formar programas que se pueden cargar (LOAD), guardar (SAVE) o ejecutar (RUN). En la modalidad "emulador de terminal" el movimiento es cartesiano, lo que significa que el papel sobre el cual se coloca al robot se divide en una cuadrícula. Cuando al plotter se le dé la instrucción de ir a (500,500), se moverá hacia estas coordenadas en vez de moverse 500 pasos en ambos sentidos. Una hoja de papel tamaño A4 posee un máximo de 2 100 coordenadas en el eje *x* y 2 970 en el eje *y*. Las instrucciones se pueden entrar tanto en modalidad absoluta como en modalidad relativa, según la instrucción MOVE lleve o no el prefijo de una A o una R. Los lápices se seleccionan con instrucciones parecidas a las del LOGO, tales como U para lápiz arriba, D para lápiz abajo y P para lápiz seleccionado.

En modalidad directa, el Penman Plotter también puede producir movimientos para gráficos de tortuga, si bien éstos son algo más complicados que los movimientos cartesianos, porque la longitud de la distancia a recorrer se debe entrar en notación hexadecimal con una $ como prefijo. Esto se debe a que los gráficos de tortuga pasan por alto el software normal que interpreta los movimientos cartesianos e investigan directamente los bits de las direcciones controladoras del interior del ordenador. Sin embargo, ésta no es la única forma en que se pueden realizar gráficos de tortuga con el Penman Plotter. El software de aplicaciones que se suministra contiene programas que permiten controlar el robot mediante LOGO.

Para controlar al plotter en la modalidad robótica se utiliza un sistema que controla directamente el Penman desde el registro de dirección de datos de la puerta para el usuario. Cada bit del registro controla un aspecto diferente de los movimientos del robot: los bits 0 y 1 y los bits 2 y 3 controlan, respectivamente, los motores derecho e izquierdo. Los bits 4 y 5 llevan a cabo las funciones generales de los motores, como encenderlos y apagarlos, mientras que el bit 6 controla el movimiento del lápiz hacia arriba y hacia abajo. El Penman también devuelve información de sí mismo al registro de datos, tal como errores posicionales y si el dispositivo ha detectado o no el borde de la página con sus sensores de luz.

Al Penman Plotter también se le pueden enviar instrucciones de texto. La gama de instrucciones y sus aplicaciones son muy similares a las que se utilizan en las impresoras/plotter existentes para una gran cantidad de máquinas personales. De hecho, la forma de los caracteres impresos de ambas guardan una notable similitud. El tamaño del texto se puede variar entre 1 y 127 mm en altura y se puede imprimir en una de cuatro direcciones. Una útil adición del Penman Plotter, de la cual no disponen las impresoras/plotter tradicionales, es la capacidad para sesgar el texto.

El manual que viene con el plotter facilita una lista completa de las instrucciones disponibles, con algunas explicaciones sobre cómo están implementadas, así como una explicación del hardware. No obstante, quizá sea un poco avanzado para el principiante, dando la impresión de que el Penman está dirigido a quienes ya poseen un conocimiento profundo de los mecanismos de su ordenador.

El Penman Plotter se puede considerar como otro adelanto en el desarrollo de una tortuga/plotter de propósito general. Utilizado correctamente, la resolución del dispositivo se aproxima a la de los plotters más convencionales que se utilizan en los estudios de diseño comerciales. Sin embargo, por el momento el software impide que el dispositivo sea una herramienta de gestión viable. El juego de programas que actualmente se suministra con el plotter está concebido básicamente como una herramienta educativa. Se espera que los usuarios escriban sus propios programas para hacer funcionar el plotter, aprendiendo, por consiguiente, los principios del software para robótica. Por el contrario, un usuario comercial no se suele preocupar especialmente por los detalles más delicados de la programación, sino que exige algo que sea fácil de usar. Como herramienta educativa, el Penman Plotter tiene un gran valor. La gama de modalidades de operación disponibles hacen que la máquina sea una propuesta interesante para las escuelas.

PENMAN PLOTTER

DIMENSIONES

335×128×55 mm

INTERFACES

RS232C, que permite su conexión a cualquier ordenador compatible con RS232

RESOLUCION

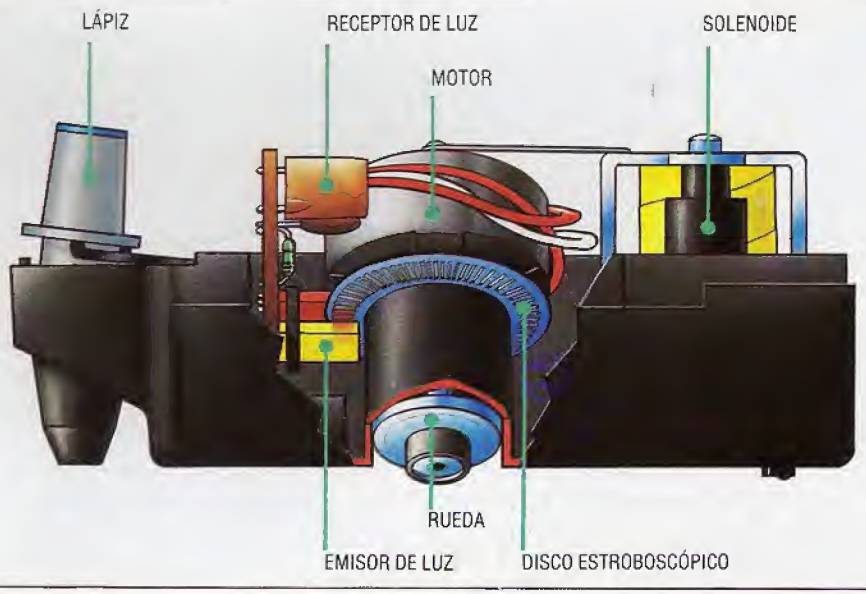
Tamaño del texto: entre 1 y 127 mm; resolución de gráficos: 0.1 mm

DOCUMENTACION

El manual proporciona muchísimos detalles técnicos que permitirán al usuario avanzado desarrollar todo el potencial del Penman, si bien carece de material de formación para el principiante

Al girar la rueda

Para lograr que el sistema de control del Penman "sepa" cuál es la posición del plotter, en el robot debe haber alguna clase de mecanismo de realimentación para contar el número de "pasos" que se han realizado. Para hacer esto, los fabricantes han colocado un delgado disco de metal alrededor de cada uno de los ejes de activación de las ruedas. El disco posee una serie de delgados cortes que van desde el centro hacia el borde. En la placa de circuitos hay montado un dispositivo que emite y recibe luz. A medida que el disco gira, se produce un efecto estroboscópico entre el emisor y el receptor en forma de una serie de impulsos. Se calcula la cantidad de éstos para determinar el número de pasos dado



Kevin Jones

Solos en el mar

Mientras navegamos hacia el Nuevo Mundo, varios eventos incidirán en la duración del viaje y en la salud de los tripulantes

El código para los eventos menores restantes se incluye en una serie de pequeñas secciones del programa que la subrutina de la línea 5500 selecciona al azar. En el capítulo anterior nos ocupamos de los primeros cinco eventos, de modo que cuando añadamos los nuevos habrá un total de 13, lo que exige cambiar el valor de RM, en la línea 46, a 13. También se debe incrementar la lista de números de línea, tras la sentencia ON X GOTO de la línea 5525, para dar cabida a las nuevas rutinas. El programa principal selecciona un evento menor de forma aleatoria, de entre su lista ampliada, para ejecutar cada semana.

```
5525 ON X GOTO 5540,5570,5570,5570,5570,5600,5700,5800,
5850,5900,5950,6000,6050
```

El primer número de línea a añadir a la sentencia ON...GOTO de la línea 5525 es el 5600: aquí comienza la rutina que se ocupa de capturar peces. Si el generador de eventos al azar determina que X sea 6, entonces la línea 5525 enviará el programa a esta rutina. Cuando muere un tripulante, la muerte no se registra en la variable CN hasta el final de la semana; pero si en el transcurso de la semana actual todo el resto de la tripulación hubiera pasado a mejor vida, obviamente no habría nadie para capturar los peces.

El programa comprueba esta eventualidad mediante la creación de un bucle en la línea 5610. Observa la matriz de la tripulación para ver si algún tripulante ha muerto durante la semana, determinando qué tasas de fortaleza se han establecido en −999. Después, la línea 5630 cuenta el número de tripulantes fallecidos y le resta el número al total. Si el resultado es menor que 1, el evento se ignorará y se devolverá el control al programa principal.

Si todavía queda vivo algún tripulante, la línea 5650 genera un número aleatorio entre 11 y 20, representando la cantidad de peces que se han cogido, expresada en kilos. Esta cantidad se suma a la carne restante en la línea 5680 y entonces el programa informa al jugador para cuánto tiempo alcanzarán las provisiones de carne. Por último, en la línea 5685, el total de carne se divide por el número de tripulantes y sus necesidades semanales, produciendo una nueva estimación de la cantidad de semanas para las que alcanzarán las provisiones.

El siguiente evento de la sentencia ON X GOTO es un temporal, que se inicia en la línea 5700. La tripulación se las apaña para recoger el agua de lluvia en barriles y reabastecer sus provisiones de agua. El programa genera un número al azar entre 11 y 20 en la línea 5735 que representa la cantidad de barriles de agua que se han recogido, y que se suma a la cantidad disponible en la línea 5750. Por último, la línea 5755 calcula e imprime la nueva estimación de cuánto tiempo durará el agua.

El octavo evento posible es que soplen vientos favorables, lo que aumentará la velocidad del barco y acortará la duración del viaje en media semana. La línea 5835 resta esta mitad al contador de semanas. La rutina de "buen tiempo" (evento número nueve) comienza en la línea 5850, la novena dirección de la línea 5525. El buen tiempo permite que la tripulación pueda tomarse un descanso para mejorar su estado de salud, lo que se simula aumentando sus tasas de fortaleza en TS(,). En la línea 5882 se prepara un bucle que recorre TS(,). Si se incrementara la tasa de fortaleza de un tripulante fallecido, esa persona "resucitaría", de modo que la línea 5884 comprueba si hay alguna tasa establecida en 0 o −999 (que significa en ambos casos la muerte) y hace caso omiso de ellos. En la línea 5886 se le suma a la tasa de fortaleza de cada tripulante vivo un número al azar entre 5 y 15.

Si el número seleccionado al azar para un evento menor es 10, el programa se dirigirá a la línea 5900, la subrutina de pérdida de medicinas. El mal tiempo ha hecho que se rompieran la mitad de los frascos de medicina, y esto evidentemente tendrá un efecto negativo en el caso de que algún tripulante se ponga enfermo. El programa comprueba primero si queda alguna medicina a bordo, examinando el primer elemento de la matriz de suministros, OA(). Si el valor está establecido en 0 o −999, no

hay ninguna medicina, de modo que este evento se ignorará. De haber alguna medicina a bordo, la línea 5925 divide la cantidad por la mitad; no obstante, puesto que los frascos no se rompen por la mitad, toma del resultado la parte entera. Después el programa imprime un mensaje informando al jugador cuántos frascos quedan.

El undécimo evento se produce cuando una ola inesperada hace que las armas se mojen y se oxiden, quedando inservibles. Las armas son una mercancía importante, que puede necesitarse para la defensa, para cambiar por alimentos o para comerciar. La subrutina que estropea las armas es similar a la que rompe los frascos de medicina. Comprueba si queda algún arma a bordo, en la línea 5955, divide por dos la cantidad en la línea 5975 e imprime la fracción entera del resultado.

El hecho de que los ratones se hayan comido la mitad de las balas de tela representa el evento número doce, que comienza en la línea 6000. La tela es el cuarto elemento de la matriz de suministros, OA(), de modo que el programa comprueba OA(4), en la línea 6005, para ver si queda algo de tela a bordo. Si se ha comprado tela, los ratones, que habitan en la línea 6025, se comen la mitad de ella y, tras el festín, se informa al jugador sobre la cantidad de tela que queda todavía.

Observe que en esta etapa hay una oportunidad de hacer alguna trampa en el juego, haciéndolo más fácil o más difícil mediante la alteración de los factores de los acontecimientos. Los límites entre los números aleatorios que se pueden seleccionar se pueden reducir o ampliar e, igualmente, se pueden aumentar o disminuir las proporciones de los suministros perdidos.

El último de los eventos menores, que, amenazadoramente, es el número 13, es la observación de un albatros. Este evento es único entre los acontecimientos menores, ya que hay dos posibilidades de que se produzca en una misma semana. Si la sentencia ON X GOTO selecciona el evento número 13, el incidente se producirá de la forma normal, como un evento menor. Si el albatros es avistado cuando la tripulación se halla a media ración de carne, se les ofrecerá la oportunidad de abatirlo para conseguir su carne. Los jugadores que conozcan el poema de Coleridge *The rime of the ancient mariner* se sentirán con toda justicia recelosos de hacerlo. En el programa principal, los eventos menores se llaman mediante el GOSUB de la línea 860; los eventos mayores comienzan en la línea 870. En la 875 se produce la segunda posibilidad de avistar el albatros. La línea comprueba si la tripulación se halla a media ración de carne, comprobando si HR(3) es igual a 0.5. De ser así, un factor aleatorio (AND RND(1)<.5) da un 50 % de probabilidades de avistar el ave. Si ésta se avista en estas circunstancias, el programa pasa a la subrutina "albatros", que comienza en la línea 6050.

Avistar un albatros trae buena suerte. En la línea 6055 se establece en "S" una variable, A$, indicando que se ha avistado el ave. Posteriormente, ésta se utilizará en el programa para reducir la posibilidad de un motín, resultado práctico de la buena suerte. El programa verifica luego, en la línea 6075, si la tripulación se está quedando sin carne, determinando si la cantidad de carne restante es igual a, o mayor que, las necesidades semanales de la tripulación multiplicadas por el tiempo que aún falte para concluir la travesía. Si las provisiones son suficientes, el ave sigue su rumbo y la línea 6130 retorna al programa principal.

En este día, 17 de julio del año de Nuestro Señor de 1589, hemos perdido otro tripulante. El pobre Robert Purkiss, mientras realizaba la segunda guardia, fue arrastrado por la borda. Dios se apiade de su alma.

Su muerte, aunque trágica, al menos ha reducido la cantidad de vituallas necesarias para cada semana. Ahora que la tripulación se halla por debajo de su número óptimo es probable que el viaje dure más tiempo de lo previsto y pienso poner a la tripulación a media ración antes de que avistemos tierra y desembarquemos.

Si las existencias de carne son escasas, se le dice al jugador que el ave pesa 10 kilos y se le pregunta si desea abatirla. La línea 6115 comprueba si, en efecto, desea intentarlo, examinando el primer carácter de la entrada. Si el jugador decide no disparar, el ave sigue su vuelo, pero si éste desea dispararle, el programa comprueba, en la línea 6133, si OA(2) está establecida en 0 (indicando que no hay ningún arma utilizable). De estar establecida en −999, las armas se habrían perdido durante la semana en curso y el programa así le informaría al jugador, mientras el ave se aleja volando.

Si hay armas a bordo, se le dispara el ave. La línea 6140 le da al jugador un 50 % de probabilidades de dar en el blanco, generando un número aleatorio entre 0 y 1. Si el número es mayor que .5 el tiro es errado y el ave se escapa; de lo contrario, se hace blanco y ésta cae en la cubierta. Si durante la semana en curso se hubieran agotado las reservas de carne, el valor correspondiente de la matriz se restablecería en 0, quitando el −999, en la línea 6160, para poder registrar las provisiones extras que proporcionaría el cuerpo del ave. La cantidad de carne de la matriz de provisiones, PA(3), se aumenta en 10 kilos y se le indica al jugador el nuevo total.

El abatimiento del albatros se señala en el poema de Coleridge mediante una interrupción del invitado a la boda:

"¡Dios te guarde, anciano marinero!
de los espíritus malignos que te atormentan.
¿Cuál es la pena que te aflige?"
"Con mi ballesta
abatí al albatros."

Ian McKinnell

En nuestro juego se le da al jugador una advertencia similar. En la línea 6162, B$ se establece en "S". Éste es el factor de mala suerte, que supera con creces a la buena suerte que trajo la línea 6055. La naturaleza exacta de esta mala suerte se revelará más adelante en el proyecto.

Módulo 7: Más eventos aleatorios

Adición al bucle principal del viaje

```
875 IF HR(3)=.5 AND RND(1)<.5 THEN PRINT CHR$(147):GOSUB
    6050
```

Subrutina más eventos al azar

```
5600 REM EVENTO 6 – CAPTURAR PECES
5605 X=0
5610 FOR T=1 TO 16
5615 IF TS(T,2)=–999 THEN X=X+1
5620 REM CONTAR FALLECIDOS ESTA SEMANA
5625 NEXT
5630 IF CN – X<1 THEN RETURN
5635 REM NINGUNA ACCION SI TODA TRIPULACION MUERTA
5640 PRINT
5645 S$="  DURANTE LA SEMANA*":GOSUB 9100
5646 PRINT:GOSUB 9200
5650 S$="UNO DE LOS TRIPULANTES CAPTURO*":GOSUB 9100
5655 X=INT (RND(1)*10)+11
5660 REM ENTRE 10 y 20 KILOS
5662 PRINT X; "KILOS DE PESCADO"
5665 PRINT:GOSUB 9200
5670 S$="AHORA TU PROVISION DE CARNE ES*":GOSUB 9100
5675 S$="SUFICIENTE PARA APROXIMADAMENTE*":GOSUB 9100
5678 IF PA(3)=–999 THEN PA(3)=0
5680 PA(3)=PA(3)+X
5685 PRINT INT(PA(3)/(CN*PN(3)));"SEMANAS"
5690 GOTO 5530
5700 REM EVENTO 7 – RECOGER AGUA
5705 PRINT
5710 S$="  DURANTE LA SEMANA*":GOSUB 9100
5715 PRINT:GOSUB 9200
5720 S$="UN TEMPORAL DE LLUVIA LLENO TUS*":GOSUB 9100
5725 S$="BARRILES DE AGUA*":GOSUB 9100
5730 PRINT:GOSUB 9200
5735 X=INT(RND(1)*10)+11
5736 REM ENTRE 10 y 20 BARRILES
5740 S$="AHORA TU PROVISION DE AGUA ES*":GOSUB 9100
5745 S$="SUFICIENTE PARA APROXIMADAMENTE*":GOSUB 9100
5748 IF PA(4)=–999 THEN PA(4)=0
5750 PA(4)=PA(4)+X
5755 PRINT INT(PA(4)/(CN*PN(4)));"SEMANAS"
5760 GOTO 5530
5800 REM EVENTO 8 – VIENTOS FAVORABLES
5805 PRINT
5810 S$="FUERTES VIENTOS CONTINUOS DURANTE TODA LA
     SEMANA*":GOSUB 9100
5815 PRINT:GOSUB 9200
5820 S$="HAS NAVEGADO A BUENA VELOCIDAD*":GOSUB 9100
5825 S$="Y LA DURACION DEL VIAJE SE*":GOSUB 9100
5830 S$="REDUCE EN MEDIA SEMANA*":GOSUB 9100
5835 EW=EW–.5
5839 GOTO 5530
5850 REM EVENTO 9 – BUEN TIEMPO
5855 PRINT
5860 S$="BUEN TIEMPO DURANTE TODA LA SEMANA*":GOSUB
     9100
5865 PRINT:GOSUB 9200
5870 S$="LA TRIPULACION SE SIENTE MAS FELIZ*":GOSUB 9100
5875 GOSUB 9200
5880 S$="  ¡Y ESTA MAS SALUDABLE!*":GOSUB 9100
5882 FOR T=1 TO 16
5885 IF TS(T,2)=0 OR TS(T,2)=–999 THEN 5888
5886 TS(T,2)=TS(T,2)+INT(RND(1)*11)+5
5888 NEXT
5889 GOTO 5530
5900 REM EVENTO 10 – PERDIDA DE MEDICINAS
5905 IF OA(1)=0 OR OA(1)=–999 THEN RETURN
5910 PRINT
5915 S$="DESCUBRES QUE LA MITAD DE TUS*":GOSUB 9100
5920 S$="FRASCOS DE MEDICINA SE HAN ROTO*":GOSUB 9100
5925 OA(1)=INT(OA(1)/2)
5930 PRINT:GOSUB 9200
5935 S$="AHORA TE QUEDAN*":GOSUB 9100
5940 PRINT OA(1);"FRASCOS SOLAMENTE"
5945 GOTO 5530
5950 REM EVENTO 11 – ARMAS OXIDADAS
5955 IF OA(2)=0 OR OA(2)=–999 THEN RETURN
5960 PRINT
5965 S$="DESCUBRES QUE LA MITAD DE TUS*":GOSUB 9100
5970 S$="ARMAS SE HAN OXIDADO*":GOSUB 9100
5972 S$="Y YA NO SIRVEN*":GOSUB 9100
5975 OA(2)=INT(OA(2)/2)
5980 PRINT:GOSUB 9200
5985 S$="AHORA TE QUEDAN*":GOSUB 9100
5990 PRINT OA(2);"ARMAS SOLAMENTE"
5995 GOTO 5530
6000 REM EVENTO 12 – PERDIDA DE TELA
6005 IF OA(4)=0 OROA(4)=–999 THEN RETURN
6010 PRINT
6015 S$="DESCUBRES QUE LOS RATONES*":GOSUB 9100
6020 S$="SE HAN COMIDO LA MITAD DE TUS*":GOSUB 9100
6022 S$="BALAS DE TELA Y AHORA*":GOSUB 9100
6024 S$="ESTAS YA NO SIRVEN*":GOSUB 9100
6025 OA(4)=INT(OA(4)/2)
6030 PRINT:GOSUB 9200
6035 S$="AHORA TE QUEDAN*":GOSUB 9100
6040 PRINT OA(4);"BALAS SOLAMENTE"
6045 GOTO 5530
6050 REM EVENTO 13 – ALBATROS
6055 PRINT:A$="S"
6060 S$="UN ALBATROS SOBREVUELA EL BARCO*":GOSUB 9100
6062 GOSUB 9200
6065 S$="ESTE ES UN BUEN PRESAGIO*":GOSUB 9100
6068 S$="Y LA TRIPULACION ESTA CONTENTA*":GOSUB 9100
6070 PRINT:GOSUB 9200
6075 IF PA(3)<(CN*PN(3)*(JL–WK+1)) THEN 6090
6080 REM NO HAY ESCASEZ DE CARNE
6085 GOTO 6122
6090 S$="SE TE ESTA TERMINANDO LA CARNE*":GOSUB 9100
6095 S$="Y EL AVE PESA 10 KILOS!*":GOSUB 9100
6100 PRINT:GOSUB 9200
6105 S$="TE GUSTARIA COGERLA?*":GOSUB 9100
6110 INPUT I$
6112 PRINT:GOSUB 9200
6115 IF LEFT$(I$,1)="S" THEN 6133
6120 S$="PROBABLEMENTE DA LO MISMO!*":GOSUB 9100
6122 PRINT:GOSUB 9200
6125 S$="EL ALBATROS SE ALEJA VOLANDO*":GOSUB 9100
6130 GOTO 5530
6133 IF OA(2)=0 OR OA(2)=–999 THEN 6180
6135 S$="HACES UN DISPARO.....*":GOSUB 9100
6138 GOSUB 9200:GOSUB 9200
6140 IF RND(1)<.5 THEN 6150
6145 S$="  .....PERO NO ACIERTAS!*":GOSUB 9100
6148 GOTO 6122
6150 S$="Y EL AVE CAE SOBRE LA CUBIERTA!*":GOSUB 9100
6155 PRINT:GOSUB 9200
6160 IF PA(3)=–999 THEN PA(3)=0
6162 PA(3)=PA(3)+10:B$="S"
6165 S$="AHORA TIENES 10 KILOS MAS*":GOSUB 9100
6167 S$="DE CARNE.....*":GOSUB 9100
6170 S$="PERO PUEDE SER QUE LA BUENA FORTUNA*":GOSUB
     9100
6172 S$="NO TE ACOMPAÑE A PARTIR DE AHORA!!*":GOSUB
     9100
6174 GOTO 5530
6180 S$="NO PUEDES...NO TIENES ARMAS*":GOSUB 9100
6190 GOTO 6122
```

Complementos al BASIC

Spectrum:

Introduzca las siguientes modificaciones:

```
875 IF HR(3)=.5 AND RND(1)<.5
    THEN CLS:GO SUB 6050
5527 IF X=6 THEN GO TO 5600
5528 IF X=7 THEN GO TO 5700
5529 IF X=8 THEN GO TO 5800
5530 IF X=9 THEN GO TO 5850
5531 IF X=10 THEN GO TO 5900
5532 IF X=11 THEN GO TO 5950
5533 IF X=12 THEN GO TO 6000
5534 IF X=13 THEN GO TO 6050
5535 PRINT:S$=K$:GO SUB 9100
5536 LET I$=INKEY$:IF INKEY$=""THEN
     GO TO 5536
6115 IF I$(1 TO 1)="S" THEN GO TO 6133
```

BBC Micro:

Introduzca las siguientes modificaciones:

```
875 IF HR(3)=.5 AND RND(1)<.5
    THEN CLS:GOSUB 6050
```

A prudente distancia

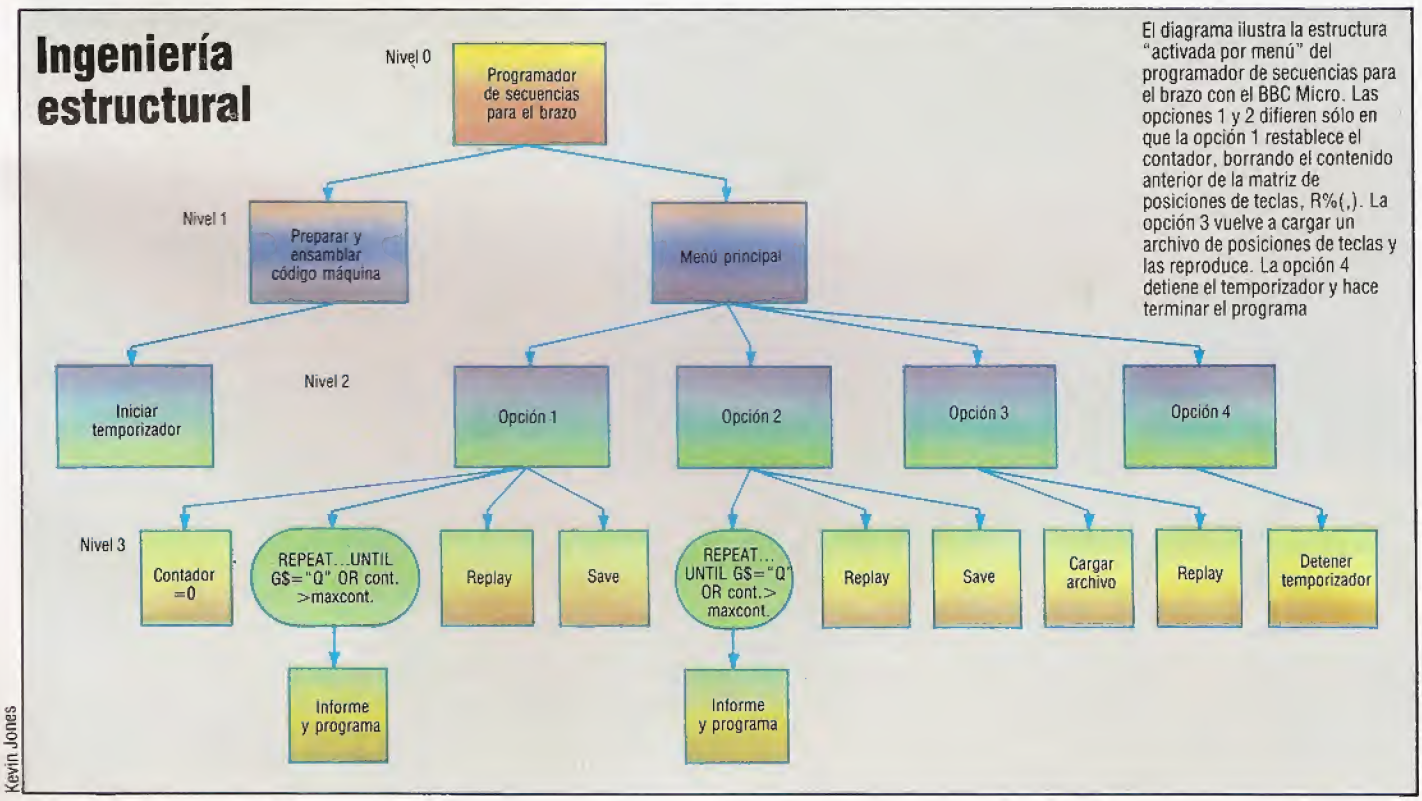
Nos corresponde diseñar el primer fragmento de software para controlar los movimientos del brazo-robot

Los cuatro motores que mueven el brazo están conectados a las cuatro líneas de datos del ordenador con las que está conectado el brazo en interface, que son también las que controlan los motores. En el caso del BBC Micro, estas cuatro líneas de datos parten de la puerta para el usuario utilizando las patillas de D0 a D3. Debido a que los servomotores se basan en un flujo continuo de impulsos que les indican qué ángulo seguir, no se los puede controlar exclusivamente desde BASIC, sino que deben ser activados mediante un programa cuña en código máquina, que envíe un nuevo grupo de señales a los motores una vez cada dieciseisavo de segundo utilizando una interrupción. (Anteriormente ya hemos explicado detalladamente los principios de este método de control.) Se puede emplear el programa de control para servomotores múltiples ofrecido con anterioridad para comprobar la operación del brazo directamente desde el teclado y, si bien no es muy sofisticado, probará cada motor de forma independiente.

Una vez que el código cuña esté ejecutándose como tarea de fondo, será posible salir del programa de exploración de teclas del BASIC o colocar números directamente en las posiciones que controlan los ángulos de los motores. La dirección de la primera posición de memoria motor-ángulo es ángulo; motor 0 se establece utilizando el valor de esta posición. La posición ángulo+1 establece la posición del motor 1, y así sucesivamente. Recuerde que el motor 0 controla el movimiento de la cintura, el motor 1 el movimiento del hombro, y el motor 3 abre y cierra el mecanismo de prensión. Si cualquier ángulo se incrementa o disminuye en una unidad, los motores lo pueden seguir fácilmente. Si se introducen cambios grandes, como de 0 a 128, entonces los motores pueden tardar unos instantes en reaccionar. Asimismo, observe cómo los efectos combinados de la inercia y "esponjosidad" del brazo hacen que éste se detenga de forma brusca, que recuerda la moda de la danza robótica.

Tras probar el brazo utilizando el sencillo programa de control, se hace evidente la necesidad de un programa más sofisticado. No obstante, decidir exactamente cómo programar el brazo probablemente sea la parte más difícil de toda la operación. Muchos robots industriales se programan moviendo la pinza manualmente a través de la secuencia que debe aprender el robot; el brazo graba de manera automática las posiciones clave de la secuencia. Lamentablemente, este método exige sofisticados elementos de hardware, como sensores de realimentación angular, de los que carece nuestro robot.

Pero incluso con equipos sofisticados, tales como los empleados para aplicaciones industriales, el



El diagrama ilustra la estructura "activada por menú" del programador de secuencias para el brazo con el BBC Micro. Las opciones 1 y 2 difieren sólo en que la opción 1 restablece el contador, borrando el contenido anterior de la matriz de posiciones de teclas, R%(,). La opción 3 vuelve a cargar un archivo de posiciones de teclas y las reproduce. La opción 4 detiene el temporizador y hace terminar el programa

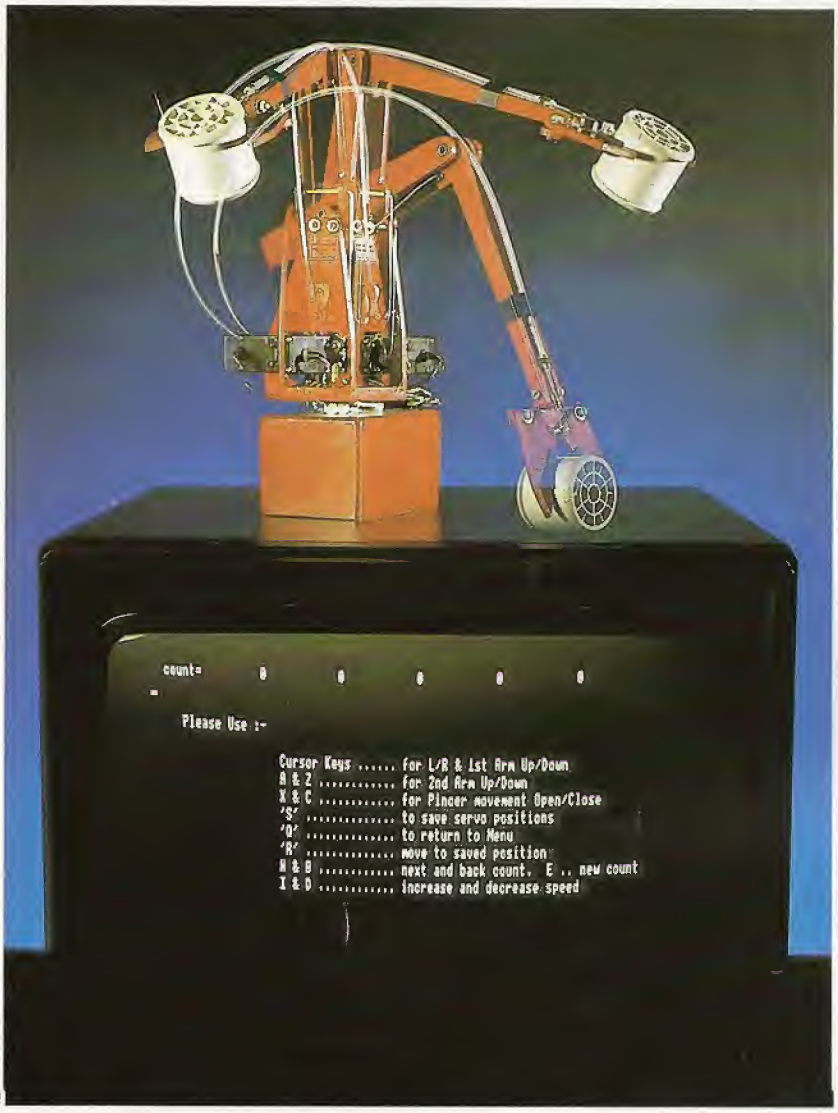
Kevin Jones

brazo podría trabajar en entornos peligrosos en los que un operador no podría físicamente programar el brazo. En consecuencia, un método alternativo consiste en controlar el brazo a distancia desde el teclado, haciéndolo pasar por una serie de movimientos y guardando las posiciones clave de la secuencia mediante la pulsación de otra tecla, y registrando los cuatro ángulos de motor que definen cada posición en una matriz. La secuencia se puede reproducir simplemente recorriendo la matriz de posiciones de teclas con un factor de retardo adecuado. Las secuencias también se pueden guardar o cargar desde cinta o disco empleando un archivo secuencial para almacenar el contenido de la matriz. Éste es el procedimiento que utilizaremos.

La primera etapa del diseño de nuestro software es, por lo tanto, asignar teclas para controlar las diversas funciones del brazo, de modo que pueda "enseñársele" una secuencia de movimientos moviéndolo a distancia desde el teclado. Se pueden reasignar las teclas del cursor del BBC Micro para que controlen el movimiento izquierda/derecha del cuerpo principal y el movimiento arriba/abajo del brazo inferior. Las teclas A y Z se utilizarán para mover el brazo superior hacia arriba y hacia abajo, y las teclas X y C para abrir y cerrar las garras de la pinza. Las teclas I y D permiten aumentar o reducir la velocidad del brazo, de modo que las maniobras delicadas se puedan llevar a cabo lentamente, mientras los movimientos de barrido amplios se realizan con rapidez. La tecla S permite almacenar la posición actual del brazo en la matriz de posiciones de teclas, e Y retorna el brazo a la última posición almacenada. La secuencia se puede realizar para introducir cambios, ya sea hacia adelante empleando la tecla N, o bien hacia atrás, pulsando la tecla B; E controla movimientos a ciertos puntos de la secuencia. Utilizando estas tres últimas teclas, se puede editar cualquier secuencia de posiciones programada.

Los procedimientos informe y programa del listado le proporcionan una lista de las funciones de las teclas, además de explorar repetidamente el teclado en busca de una entrada.

Los otros procedimientos, replay y save, permiten reproducir o guardar en cinta o disco la secuencia retenida en la matriz de posiciones de teclas, R%(,), mientras que cargararchivo reasigna los valores de R%(,) utilizando valores previamente guardados en un archivo secuencial. Por tanto, tomado como un todo, el programa nos ofrece un método cabal para programar, editar y reproducir secuencias de movimientos del brazo.



Ian McKinnell

Recogiendo las piezas
Comenzamos el proyecto del brazo-robot considerando una tarea determinada: recoger y trasladar un objeto del tamaño y peso de una carrete de hilo de coser. El programador de secuencias nos permite programarlo para que realice esta tarea, guardando los movimientos, si fuera necesario, en un archivo que se puede volver a cargar y reproducir

Un movimiento más uniforme

La rutina en código máquina para manipulación de interrupciones controla los ángulos de los servomotores analizando cuatro posiciones, que comienzan en la dirección ángulo. Cada una de estas posiciones contiene un valor que corresponde al ángulo que debe asumir cada motor. Una de las razones por las cuales el programa de prueba original produce movimientos discontinuos es que el programa en BASIC coloca (POKE) valores directamente en estas posiciones. Los grandes cambios en un ángulo producen un movimiento violento del motor, dado que éste intenta alcanzar la nueva posición con la mayor rapidez posible. Para conseguir que el brazo se desplace de forma más uniforme, deberíamos tratar de ajustar los valores de las cuatro posiciones ángulo sólo en pequeñas cantidades. Por este motivo se utiliza un segundo grupo de posiciones, que comienzan en nuevapos%, para aceptar los cambios de ángulo desde el programa en BASIC. Se puede añadir luego al programa una corta rutina en código máquina para aumentar o disminuir los valores de las cuatro posiciones ángulo (en pasos de una unidad) hasta que concuerden con los valores de la correspondiente posición nuevapos%. El procedimiento moverservo simplemente llama al programa en código máquina que hace esto.

La introducción de esta nueva rutina ofrece una ventaja adicional. Mediante la inserción de un bucle de demora dentro de este trozo de código máquina, podemos retardar o acelerar la velocidad a la cual se mueve el motor desde su antigua posición a su nueva posición, simplemente alterando el valor que da por terminado el bucle de demora. La posición &81 almacena este factor de retardo, que se puede alterar desde el BASIC al comienzo del procedimiento replay, permitiendo implementar a diferentes velocidades las secuencias de movimientos programados previamente. El código máquina para esta nueva rutina comienza en la línea 2000 del listado que ofrecemos.

Prog. secuencias BBC Micro

```
10 REM ********************************
11 REM ********************************
12 REM **                              **
13 REM **          PROGRAMADOR         **
14 REM **     SECUENCIAS BRAZO BBC     **
15 REM **                              **
16 REM ********************************
17 REM ********************************
20 :
25 MODE 3
30 PROCpreparacion
40 CLS:PRINT TAB(5,6)"Te gustaria?:"
50 PRINT TAB(25,10)"1 ..... Programar nueva secuencia para brazo"
60 PRINT TAB(25,12)"2 ..... Añadir movimientos al programa"
70 PRINT TAB(25,14)"3 ..... Reproducir un archivo"
80 PRINT TAB(25,16)"4 ..... Salir del programa"
90 GS=GET$:IF GS="1" THEN contador=0
100 IF GS="1" OR GS="2" THEN PROCinforme:PROCprograma:PROCreplay:PROCsave
110 IG GS="3" THEN PROCcargararchivo:PROCreplay
120 IF GS="4" THEN CLS:PROCfintemporizador:END
130 GOTO 40
140
150
160 DEF PROCinforme:CLS
170 PRINT TAB(5,5)"Por favor utiliza :"
180 PRINT TAB(20,7)"Teclas cursor...para I/D y 1.er brazo Ar/Ab"
190 PRINT TAB(20)"A & Z ..... para 2do brazo Ar/Ab"
200 PRINT TAB(20)"X & C ..... para movimiento Abrir/Cerrar pinza"
210 PRINT TAB(20)"'S' ..... para guardar posiciones servos"
220 PRINT TAB(20)"'Q' ..... para regresar al Menu"
230 PRINT TAB(20)"'R' ..... mover a posicion guardada"
240 PRINT TAB(20)"N & B ..... siguiente y anterior contador.
                                E .. nuevo contador"
250 PRINT TAB(20)"I & D ..... aumentar y reducir velocidad"
260 PRINT TAB(2,2)"contador="contador"    ";:FOR I%=0 TO 3:PRINT R%
    (contador,I%);"    ";:NEXT:PRINT
270 ENDPROC
280
290
300 DEF PROCprograma:?&81=1
310 REM reasignar teclas movimiento cursor
320 *FX4,1
330 DX=2
340 REPEAT :REM explorar teclas
350   IF INKEY(-38)THEN DX=DX+1   :REM "I"
360   IF INKEY(-51)THEN DX=DX-1   :IF DX<1 THEN DX=1:REM "D"
380   IF INKEY(-26)THEN nuevapos%?0=(nuevapos%?0)+DX:REM "FLECHA IZQUIERDA" servo 0 derecha
390   IF INKEY(-122) THEN nuevapos%?0=(nuevapos%?0)-DX: REM "FLECHA DERECHA" servo 0 izquierda
400   IF INKEY(-58)THEN nuevapos%?1=(nuevapos%?1)+DX: REM "FLECHA ARRIBA" servo 1 arriba
410   IF INKEY(-42)THEN nuevapos%?1=(nuevapos%?1)-DX: REM "FLECHA ABAJO" servo 1 abajo
420   IF INKEY(-66)THEN nuevapos%?2=(nuevapos%?2)-DX: REM "A" servo 2 arriba
430   IF INKEY(-98) THEN nuevapos%?2=(nuevapos%?2)+DX: REM "Z" servo 2 abajo
440   IF INKEY(-67) THEN nuevapos%?3=(nuevapos%?3)+DX*3: REM "X" servo 3 abrir pinza
450   IF INKEY(-83) THEN nuevapos%?3=(nuevapos%?3)-DX*3: REM "C" servo 3 cerrar pinza
460   IF INKEY(-52) THEN FOR I%=0 TO 3:nuevapos%?I%=R%(contador,I%): NEXT:REM "R" reproducir posicion
470   IF INKEY(-86)THEN contador=contador+1:IF contador>maxcontador THEN contador=0
480   IF INKEY(-101)THEN contador=contador-1:IF contador<0 THEN contador=maxcontador
490   IF INKEY(-35)THEN PRINTTAB(10,20)"DAR VALOR PARA contador    ";: INPUT contador :REM "E" entrar nuevo
      contador
500   IF INKEY(-82)THEN FOR I%=0 TO
      Ningunservo%:R%(contador,I%)=nuevapos%?I%:NEXT:contador=contador+1:REM "S" guardar posicion en
      contador actual
510   IF contador>ultimomaxcont THEN ultimomaxcont=contador
520   IF antcont<>contador THEN PRINTTAB(2,2)"contador="contador"    ";:FOR I%=0 TO 3:PRINT
      R%(contador,I%);"    ";:NEXT:PRINT
530   antcont=contador
540 FOR I%=0 TO 3:IF nuevapos%?I%>200 THEN nuevapos%?I%=200
550 IF nuevapos%?I%<10 THEN nuevapos%?I%=10
560 NEXT
570 PROCmoverservo
580 GS=INKEY$(1):UNTIL G$="Q"OR contador>maxcontador
590 *FX4,0
600 ENDPROC
610
620
630 DEF PROCreplay
640 REPEAT
650 PRINTTAB(11,23)"Quieres reproducir movimientos? (S/N) R para repetir"
660 GS=GET$
670 UNTIL GS="S" OR GS="N" OR GS="R"
680 IF GS="N" THEN ENDPROC
690 IF GS<>"R" THEN INPUTTAB(11,22)"Que periodo de demora entre respuesta? (1 a 255) despues <return>";dly%:IF
    dly%>255 OR dly%<1 GOTO 690
700 ?&81=dly%
710 FOR NumPos=0 TO ultimomaxcont:PRINTTAB(19,2)"Num. de posicion secuencial   =   ";NumPos;"    ";FOR
    servo%=0 TO Ningunservo%:nuevapos%?servo%=R%(NumPos,servo%):NEXT:PROCmoverservo:NEXT
720 GOTO 640
730
740
750 DEF PROCmoverservo
760 CALLmoverservo :REM version codigo maquina
820 ENDPROC
830
831
840 DEF PROCsave:CLS
850 PRINT TAB(15,10)"Te gustaria guardar la secuencia en un archivo? S/N  ";:IF GET$<>"S" THEN ENDPROC
860 INPUT TAB(15,12)"Por favor nombra el Archivo a guardar : ";archivo$ ' ' :archivo$="D."+archivo$
870 PROCfintemporizador
880 X=OPENOUT archivo$:PRINT#X,ultimomaxcont:FOR A=0 TO ultimomaxcont:FOR B=0 TO
    Ningunservo%:PRINT#X,R%(A,B):NEXT:NEXT
890 CLOSE#X
900 PROCiniciartemporizador
910 ENDPROC
920
930
940 DEF PROCcargararchivo:CLS
950 INPUT TAB(15,10)"Por favor entra el nombre del archivo a cargar:";archivo$:archivo$="D."+archivo$
960 PROCfintemporizador
970 X=OPENIN archivo$:INPUT #X,contador:ultimomaxcont=contador:FOR A=0 TO contador:FOR B=0 TO
    Ningunservo%:INPUT#X, R%(A,b):NEXT:NEXT
980 antcont=0
990 CLOSE#X
1000 PROCiniciartemporizador
1010 ENDPROC
1020
1030
1040 DEF PROCpreparacion
1050 contador=0:Ningunservo%=3:REM NUM. DE MOTORES-1
1060 ultimomaxcont=0  :antcont=0  :dly%=10
1070 maxcontador=100
1080 DIM R%(maxcontador,Ningunservo%),nuevapos% 8,temporizador% 12, leer% 12
1090 PROCassembletime
1100 ENDPROC
1110
1120
1130 DEF PROCassembletime
1140 REM ********************************************************
1150 REM                    Establecer temporizador etc.
1160 REM ********************************************************
1170 osbyte=&FFF4
1180 A%=&97  :X%=&62  :Y%=&FF
1190 CALL osbyte:REM establecer puerta B para salida
1200 DIM p%(8)
1210 DIM temporizador% 12,leer% 12
1220 xtemporizador=temporizador% MOD 256
1230 ytemporizador=temporizador% DIV 256
1240 xleer=leer% MOD 256
1250 yleer=leer% DIV 256
1260 PROCinicial
1270 FOR I%=angulo TO angulo+8:angulo?I%=128:NEXT
1280 t=.02  :REM seg entre impulsos
1290 tiempo%=&FFFFFFFF-(t*100)+1
1300 temporizador%?4=&FF   :REM cargar byte mas alto
1310 !temporizador%=tiempo%:REM establecer temporizador, habilitar eventos
1320 PROCiniciartemporizador
1330 ENDPROC
1340
1350 DEF PROCiniciartemporizador
1360 *FX14,5
1370 A%=4   :X%=xtemporizador   :Y%=ytemporizador   :CALL &FFF1
1380 ENDPROC
1390
1400 DEF PROCfintemporizador
1410 *FX13,5
1420 ENDPROC
1430
1440
1450 DEF PROCinicial
1460  REM ********************************************************
1470  REM                    Ensamblar el codigo maquina
1480  REM ********************************************************
1490 DIM espacio% 500
1500 FOR C=0 TO 3 STEP 3
1510 paginacero=&70   :REM libre para usuarios
1520 puertab=&FE60   :osword=&FFF1
1530 P%=espacio%
1540 angulo=P%  :P%=P%+8  :REM potencialmente 8 motores
1550 tabla=P%  :P%=P%+256  :REM 256 longitudes de impulso posibles
1560 FOR I%=tabla TO tabla+&100:?I%=&FF:NEXT
1570 lowtabla%=tabla MOD 256
1580 hightabla%=tabla DIV 256
1590 ?paginacero=lowtabla%  :paginacero?1=hightabla%
1600 [OPT C
1610 .manipuladoreventos
1620 PHP:PHA:TYA:PHA:TXA:PHA
1630 LDA #&04
1640 LDX #xtemporizador
1650 LDX #ytemporizador
1660 JSR osword
1670 LDY #&0
1680  /Empezar impulso, para algunos motores seria posible empezar
1690  /antes de llenar la tabla y reducir asi el bucle de espera de abajo
1700 LDA #&FF  :STA puertab
1710  /llenar tabla con excepciones
1720 LDX #&7  :LDA #FF  :CLC /preparar patron bits
1730 .excepciones
1740       ROR A  :PHA /patron bits
1750       LDY angulo,X /tomar despl. corresp. a angulo motor X
1760       AND (paginacero),Y /conservar patron bits existente
1770       STA (paginacero),Y /pero modificado para motor X
1780       PLA :DEX
1790       BPL excepciones
1800 /ahora la tabla esta cargada, se llenara en poco tiempo
1810 LDY #&60
1820 .esperar      DEY
1830       BNE esperar
1840 LDA #&FF /todos los impulsos encendidos
1850 LDY #&0
1860 .bucle AND (paginacero), Y/pero desenmascarar con cada
1870       STA puerta b /de la tabla de uno en uno
1880       INY
1890       BNE bucle
1900 LDX #&7  :LDA #&FF
1910 .borrar
1920       LDY angulo,X /borrar otra vez todas las excepciones
1930       STA (paginacero),Y
1940 DEX
1950 BPL borrar
1960 /todos los impulsos han de haber acabado ya
1970 PLA:TAX:PLA:TAY:PLA:PLP
1980 RTS
1990 ,
2000 .
2010 .moverservo /.....movimiento uniforme cod. maq. .....
2020 .Bucle2
2030 LDY #0
2040 LDY #0
2050 .Bucle1
2060   LDA nuevapos%,X        /  nueva posicion para servo x
2070   CMP angulo,X           /  es la antigua posición la misma?
2080   BEQ movido             /  if=no hacer nada
2090     BSC sumar            /  if>sumar
2100       DEC angulo,X   /  else sacar uno
2110       JMP incrementarx
2120     .sumar
2130       INC angulo,X
2140       JMP incrementarx
2140     .movido
2150       INY
2170   .incrementarx
2180   INX  :CPX #4
2190   BNE Bucle1             /  he comprobado los 4 servos?
2200 JSR esperarmotores
2210 CPY #4
2220 BNE Bucle2               /  continuar si no han terminado todos
2230 RTS                      /  al llegar aqui todos los servos en pos correcta
2240
2250
2260 .esperarmotores
2270 TXA:PHA:TYA:PHA
2280 LDY #&FF  :LDX &81
2290 .esperando
2300   DEY   :NOP  :NOP
2310   BNE esperando          /  bucle 8 ciclos reloj, o sea 1024 ciclos
2320 LDY #&FF
2330 DEX
2340 BNE esperando
2350 PLA:TAY:PLA:TAX
2360 RTS
2370
2380 ]
2390 NEXT
2400 !&220=manipuladoreventos OR (!&220 AND &FFFF0000)
2410 ENDPROC
```

Lenguaje estructural

En PASCAL es posible definir procedimientos y funciones para complementar aquellos que ya están incorporados en el lenguaje

Antes de dar por finalizado nuestro informe sobre las estructuras de datos del PASCAL, echaremos una mirada a algunos mecanismos para estructurar programas. Ello se realiza definiendo nuestros propios procedimientos y funciones para complementar aquellos que ya se hallan incorporados en el lenguaje PASCAL, tales como el procedimiento write y la función sqrt. Pero, antes de definir algunos por nuestra propia cuenta, veamos por qué write es un procedimiento y no una función, y por qué sqrt es una función y no un procedimiento. Cuando decimos, por ejemplo:

```
write ('Hola!')
```

esperamos que la serie de caracteres proporcionados como el parámetro simple aparezca en el archivo de salida estándar. En otras palabras, write identifica un subprograma que sabe cómo enviar datos en forma de una corriente de caracteres al dispositivo de salida (VDU). Pero ¿qué haría la siguiente "sentencia"?

```
sqrt(256)
```

La respuesta, por supuesto, es que generaría un mensaje de error de tiempo de compilación: no es una sentencia legal. Por otra parte:

```
write (sqrt(256))
```

no sólo sería legal, sino que produciría algo así como 1.60000E+01 en la pantalla. Pregúntese a sí mismo la razón, y por qué parece ser tan inevitable y obvio; si puede darse cuenta de los motivos, no albergará ningún tipo de duda sobre la diferencia fundamental entre un procedimiento y una función.

El identificador write invoca a un proceso que envía datos a una corriente de salida y es un procedimiento al cual se puede llamar por su nombre. Otro procedimiento del PASCAL es page (F), que se emplea para comenzar una nueva página en el archivo de textos F. El identificador sqrt, sin embargo, siempre debe proporcionarse con un "argumento" numérico simple y simplemente devuelve un resultado: el número real que es la raíz cuadrada de su argumento. De modo que la primera distinción entre procedimientos y funciones es que no existe una sentencia de función como tal. Así como escribir 16 de forma aislada no significa nada (no es más que un valor), las expresiones tales como sqrt(X/3) u odd(N) no poseen ningún significado propio. Los identificadores de funciones se comportan como variables que no han sido inicializadas, pero cuyo valor se calcula cada vez que se utiliza su nombre en una sentencia de programa. El resultado de valor simple que se devuelve es determinado a partir de una inspección del valor (o valores) actual del o los argumentos de la función; es decir, el resultado es una función del o los argumentos.

Los procedimientos, por el contrario, no devuelven un valor y, por tanto, no se pueden utilizar en expresiones. Esto no es sorprendente. La sentencia N:=WriteLn demuestra a las claras su ilegalidad, tal como lo sería LET N=PRINT en BASIC.

La tarea de escribir programas muy extensos en PASCAL se simplifica gracias a la capacidad para otorgar nombres a nuestros propios procedimientos y funciones y controlar su accesibilidad y la relación entre ellos. Tal como sucede con los nombres que se les puede dar a las variables, la relativa carencia de restricciones respecto a los identificadores significa que para nuestros subprogramas podemos elegir nombres útiles que tengan algún significado. Consideremos el siguiente problema:

```
PROGRAM Incompleto(input,output,archivodatos);
                  {declaraciones..}
BEGIN
  Abrir(archivodatos);
  WHILE NOT EoF (archivodatos)DO
    BEGIN
      read (archivodatos,item);
      Procesar(item)
    END
END.
```

Aun cuando todavía no hayamos visto el tratamiento de archivos, usted no tendrá ninguna dificultad en comprender qué es lo que hace el programa. En la parte de declaraciones, declararíamos un procedimiento para localizar un archivo de datos en el almacenamiento de apoyo y abrirlo para leerlo (Abrir) y otro para tratar cada elemento de datos de la forma adecuada (Procesar). El programa utiliza, asimismo, dos de los identificadores predefinidos del PASCAL. El procedimiento read (que hasta ahora hemos utilizado para leer entradas de texto) se puede emplear, como podemos ver, para leer datos, estructurados o no, de un archivo del tipo apropiado. La función booleana EoF (*End of File*: final del archivo) devuelve el valor "verdadero" tras haberse leído el último registro del archivo de su argumento. Nuevamente, obviando el identificador del archivo, pasa por defecto a la entrada del archivo estándar. Sólo para los archivos de texto, la función EoLn (*End of Line*: final de línea) devuelve "verdadero" cuando se ha leído el último carácter de una línea. Veamos cómo se aplica todo esto cuando definimos nuestros propios subprogramas.

En esencia, existe poca diferencia entre programas, procedimientos y funciones; todos son módulos que poseen sus propias descripciones de datos locales y cuerpos de sentencias. La única variación de sintaxis se produce en el encabezamiento. Un encabezamiento de programa, como ya hemos visto, se compone de la palabra reservada PROGRAM, un identificador del usuario que da nombre

al programa y una lista de parámetros que identifican los archivos con los cuales se comunicará el programa. El encabezamiento de un procedimiento sustituye la palabra reservada por PROCEDURE y amplía la "lista formal de parámetros" para incluir los identificadores del tipo de cada parámetro de datos. De modo que, por ejemplo:

```
PROCEDURE Lineas (NumLineas : integer);
  VAR
    N : integer;
BEGIN
  FOR N:=1 TO NumLineas DO
    WriteLn
END;
```

El END del procedimiento va seguido de un punto y coma, y no de un punto, como el END del final de un programa. A NumLineas, el identificador formal del parámetro, se le pasa el valor concreto en la sentencia de llamada:

```
Lineas (5)
```

Este procedimiento indudablemente trivial es, sin embargo, de mucha utilidad si con frecuencia deseamos dejar varias líneas en blanco (cinco, en este caso) para que la salida sea más clara, y ahorrarnos (a nosotros y al compilador) el tener secuencias más bien aburridas de sentencias WriteLn separadas por puntos y coma. Este procedimiento de propósito general le permitirá dejar en blanco tantas líneas como le sea necesario, de modo que Lineas(10) producirá 10 líneas en blanco, Lineas(20) producirá 20, y así sucesivamente.

Este ejemplo es, asimismo, una buena ilustración de la seguridad del PASCAL. La variable de control del bucle FOR siempre se *debe* declarar como una variable local. No era posible, por ejemplo:

```
For NumLineas := 1 TO NumLineas DO...
```

Aunque esto sería aceptable en un programa principal, no se puede garantizar la seguridad del bucle si un controlador de bucle no es local o "relativamente global". Quizá usted alguna vez se pasó horas depurando un programa en BASIC que contenía una sentencia como ésta:

```
300 FOR N=1 TO T
400 GOSUB 2000
500 NEXT N
```

¡para descubrir, finalmente, que una subrutina remota, tal vez llamada indirecta y condicionadamente, utilizaba N para algún otro fin! El PASCAL prohíbe expresamente un comportamiento tan incontrolado como éste, y ayuda a ahorrar el tiempo desperdiciado en el desarrollo. Todo identificador declarado dentro de un bloque posee una "región" definida que se extiende a través de ese bloque. Sin embargo, su esfera de acción o su "ámbito" (la parte del bloque desde la cual es accesible) sólo se extiende desde el punto de su declaración hasta el final de ese bloque, y esto se puede limitar aún más mediante la redefinición. En el ejemplo anterior, la N a la que se alude en el procedimiento Lineas es, naturalmente, el entero declarado localmente, y esta declaración anula temporalmente cualquier otra relativamente global durante cada activación del procedimiento.

Refiriéndonos al programa principal (Ambito) del ejemplo, la región de las variables globales N y X es la totalidad del programa. La esfera de acción de X comprende desde su declaración hasta el final del programa, excluyendo cualquier otro sitio dentro del procedimiento A. Ello se debe a que se define otra X (SET OF char) como un identificador de tipo y su región es el bloque de A. De modo similar, dentro de B, N alude al parámetro formal de B, y no a la variable global.

Aunque la región tanto de A como de B es la totalidad del programa, la esfera de acción de B sólo empieza en su punto de definición y, por tanto, si bien usted podría utilizar A en el procedimiento B, B es invisible para A. Ello refuerza la lógica cáustica del PASCAL: ¡no se puede ejecutar ningún programa hasta que no haya sido escrito! Esto explica, asimismo, por qué las definiciones y las declaraciones *deben* respetar el orden CONST, TYPE, VAR, seguidas de las definiciones de procedimiento y función ordenadas de acuerdo a las exigencias estructurales del programa. Muchos compiladores de PASCAL son "de un solo paso" (sólo leen el código fuente una vez), y esto no sería posible si no se respetara este orden lógico.

El esfuerzo adicional que supone el tener que declarar los datos locales a cada procedimiento se recompensa con creces. Se obtiene modularidad gracias al paso de todos los valores de datos como parámetros a procedimientos, y si bien quizá usted vea programas en PASCAL que utilizan procedimientos sin ninguna lista de parámetros, accediendo globalmente a todos los datos, es recomendable evitar estrictamente esta práctica. En realidad el hecho de que permita esta clase de abusos se puede considerar como uno de los puntos débiles del PASCAL. Observe que, así como cualquier referencia a X dentro de A significa el tipo X, la utilización de N dentro de N alude a su parámetro formal local y el entero global N queda temporalmente inaccesible. Aparte de la seguridad inherente, esto permite que un equipo de programadores trabajen juntos en un proyecto grande sin tener que preocuparse por conflictos de identificadores.

Muchísimo alcance

```
PROGRAM Alcance;
VAR
    N    : integer;
    X    : real;
PROCEDURE A(Y:real);
  TYPE
    X=SET OF char;
  {etc.}
PROCEDURE B (N:integer);
  {etc.}
BEGIN {Programa principal
  —Alcance}
  "
  ...
  A(succ(N)/3);
  B(N);
  A(X)
    {etc.}
END.
```

En el programa Alcance, que esbozamos aquí, las regiones y las esferas de acción de las diferentes variables y procedimientos se reflejan en la siguiente tabla:

Proc/Var	Región	Ambito (alcance)
A	principal	A, B, princ.
B	principal	B, principal
N (global)	principal	A, principal
X (real)	principal	B, principal
X (en proc A)	A	A
N (en proc B)	B	B

La llamada a B pasa el valor de la global N como un parámetro real, que resulta tener el mismo nombre que en B pero que es una variable completamente separada. Ahora conviene puntualizar:

- Al entrar en un bloque se realiza una copia local del valor pasado.
- Cualquier cambio de este "parámetro de valor" dentro de su bloque no incide en el parámetro real que se pasa desde el punto de llamada.
- El valor pasado puede ser cualquier expresión del tipo correcto.

Naturalmente, cualquier sentencia de procedimiento debe listar los parámetros reales de la llamada, y éstos deben corresponderse con la lista de parámetros formales del encabezamiento en número, posición y tipo. El mecanismo de paso de parámetro por valor se puede considerar como la inicialización del identificador del parámetro formal al valor, cualquiera que fuera, especificado en la sentencia de llamada. De modo que:

```
Lineas (succ(N+vacio)DIV 2)
```

implica la siguiente sentencia de asignación al entrar en Lineas:

```
NumLineas := succ(N+vacio)DIV 2
```

Teniendo presente esta forma segura de pasar valores, quizá le interese imaginar cómo podríamos escribir un procedimiento para procesar algunos datos cuando se requiriese alterar los valores de esos datos.

De base a base

El programa ValorBase ilustra el uso de procedimientos con parámetros de valor. Se puede seleccionar cualquier base entre 2 (binaria) a 16 (hexadecimal), y los números decimales entrados desde el teclado se visualizarán en la notación apropiada. Por ejemplo, 32767 (MaxInt en varios compiladores pequeños de PASCAL) se convertiría en 7FFF en hexadecimal, 111111111111111 en binario o 77777 en octal.

Intente implementar estas modificaciones:

- Para manejar la representación negativa, debemos convertir el número a su complemento a dos. A nivel máquina, esto se realiza negándolo y sumándole uno. ¿Puede pensar alguna forma sencilla de hacer esto en PASCAL?
- ¿Le interesaría, tal vez, efectuar conversiones en dirección contraria? De ser así, podría utilizar el programa como modelo para tomar un número de cualquier base (entre 2 y 16) y producir el resultado en forma decimal. ¿Podría incorporar esto como un procedimiento en el programa anterior?

Una pista: piense en los datos y en la forma de enlazarlos.

```
PROGRAM       ValorBase     (input,output);
        {convierte numeros decimales a CUALQUIER base en
        la escala de binaria a hexadecimal}
CONST
        Columnas  = 79; {para pantalla/impresora}

TYPE
        byte      = 0..255;
        cardinal  = 0..MaxInt;

VAR
        numero,
        base      : integer;
        Harto,
        legal     : boolean;

        {11111111111111111111111111111111}

PROCEDURE  EscribirDigito  (digito : byte);
    {valor de lista [contador] pasada a digito}
  BEGIN
    IF digito IN [0..9]
      THEN
        write (digito : 1)
      ELSE          {representar como A .. F}
        CASE digito OF
          10      : write ('A');
          11      : write ('B');
          12      : write ('C');
          13      : write ('D');
          14      : write ('E');
          15      : write ('F');
        END       {CASE}

  END;  {EscribirDigito}
        {11111111111111111111111111111111}

PROCEDURE  Imprimir     ( N       : cardinal;
                          base    : byte);
  CONST {todos estos datos son locales a Imprimir}
        MaxDigitos=32;

  TYPE
        limites   = 1..MaxDigitos;
  VAR
        lista     : ARRAY [limites] OF byte;
        indice    : limites;
        contador  : byte;
  BEGIN
    contador := 0;

    REPEAT
        contador := succ (contador);
        lista [contador] := N MOD base;
        N := N DIV base

    UNTIL N=0;
              {imprimir comenzando por BmsS:}
    FOR contador := contador DOWNTO 1 TO
      EscribirDigito (lista [contador])

  END;  {Imprimir}
        {11111111111111111111111111111111}
BEGIN   {ValorBase – Programa principal}

  REPEAT
    WriteLn ('Elegir una base numerica : ');
    WriteLn ('2... 16                  ':Columnas);
    WriteLn ('(con cualquier otra se sale)':Columnas);
    write ('Base?');
    read (base);
    legal := base IN [2 .. 16];

    IF legal THEN
      BEGIN
        write ('Numero ( 0 cambia base ) ? ' );
        read (numero) ;
        Harto := numero < = 0;

        WHILE NOT Harto DO
          BEGIN
            write (numero : Columnas DIV 2, ' a base ', base :
            1, 'es ');
            Imprimir ( numero, base );
            WriteLn;
            write ( 'Numero ? ');
            read (numero );
            Harto := numero<=0;
          END
      END

  UNTIL NOT legal

END.
```

Hacia el siglo XXI

Finalizamos nuestra serie sobre educación con una reflexión acerca de cómo la informática puede revolucionar los métodos educativos

El futuro de la informática educativa presenta un agudo contraste con su pasado. Un hardware más barato, un software perfeccionado y el desarrollo tecnológico se están combinando para cambiar el rostro de la educación tal como la conocemos hoy. Lejos de ser el pariente pobre del mercado de consumo, la educación se está manifestando rápidamente como una de las mayores áreas de desarrollo de una nueva tecnología. La causa de esta súbita mutación de la imagen de la informática educativa no es difícil de entrever. En los primeros días de la revolución tecnológica, la educación fue claramente discriminada por las empresas de hardware y software en favor del más lucrativo mercado de consumo. Esta situación se está modificando rápidamente, en la medida en que los productores empiezan a comprender la importancia de vender en las escuelas e inculcar una conciencia de las marcas comerciales entre los niños, que con el tiempo serán consumidores por derecho propio.

El efecto inmediato de esta actitud es el abaratamiento y la mayor disponibilidad de hardware. En 1985 Apple ha iniciado una campaña vendiendo el Macintosh en las escuelas con un descuento del 30 %. Atari ha seguido el camino marcado por Apple diseñando su nueva gama de micros de 32 bits ST teniendo en mente el mercado educativo,

Alan Adler

Centro de aprendizaje
Los ordenadores pueden revolucionar nuestro concepto de la educación al colocar al niño en el centro del proceso del aprendizaje y darle el control de los diferentes recursos para la enseñanza que la nueva tecnología pondrá a su disposición. La tecnología "vía satélite" y los sistemas de conexión en red permitirán que el estudiante se comunique con bases de datos de conocimientos de diferentes países y culturas de todo el mundo. Las escuelas podrían ceder paso a "centros de supervisión del aprendizaje", donde los niños se reunirían para analizar su trabajo, la mayor parte del cual se podrá llevar a cabo en el hogar. Los sistemas de video interactivo pueden constituir una base para el desarrollo de software sumamente complejo y se pueden utilizar robots para demostraciones físicas de conceptos abstractos. Además, el estudiante podrá acceder a los sistemas locales de ordenadores centrales, utilizando un micro a modo de terminal inteligente, y un sistema de este tipo podría supervisar el intercambio de datos, la comunicación alumno-alumno, y ofrecer un mayor poder de proceso cuando ello fuera necesario

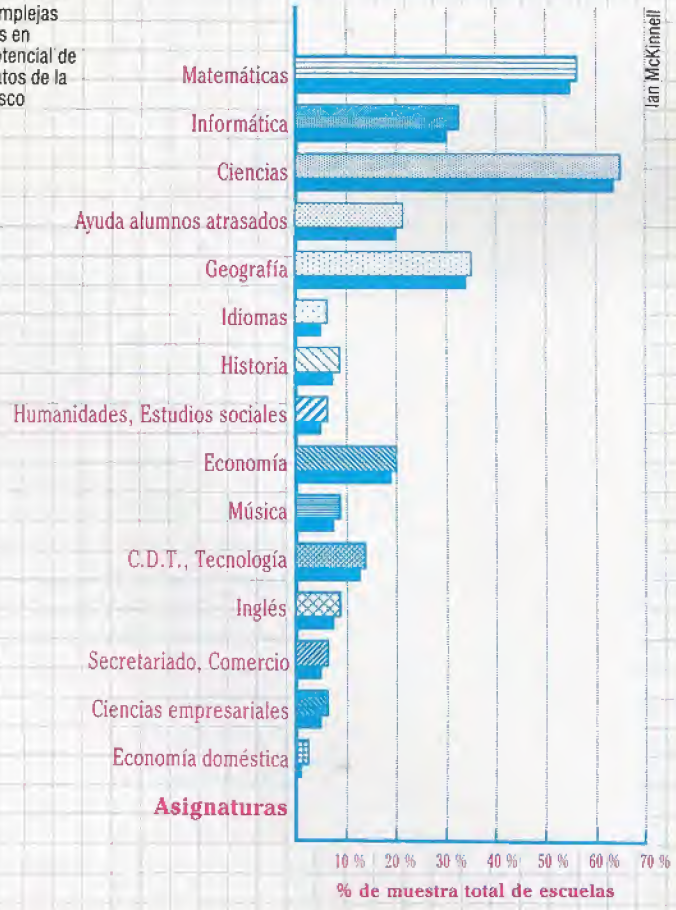
ofreciendo la opción de adquirir una máquina con LOGO en lugar de BASIC, suministrado en ROM.

Aparte del sentido comercial de penetrar el mercado educativo, ha surgido, asimismo, una segunda generación de usuarios de ordenadores que demandan usos más serios para sus micros. El desarrollo de software que previamente hubiera estado dirigido de modo exclusivo a las escuelas, de pronto está hallando un mercado más amplio, con lo cual la inversión en software educativo "serio" se convierte en una mejor propuesta comercial. La escasez de hardware y el escaso desarrollo de software ya no constituyen, por lo tanto, un obstáculo grave para la informática educativa; siendo así, ¿qué depara el futuro? El uso de ordenadores en la educación puede ahora avanzar en dos direcciones: la extensión y mejora de las técnicas existentes, y la introducción de métodos totalmente nuevos.

Seymour Papert ya ha señalado algunas ampliaciones que le gustaría introducir en el concepto del LOGO, utilizando el potencial de máquinas como el Macintosh. Papert desea ver escrito en el lenguaje un "microcosmos físico" que permita al niño jugar con las leyes del movimiento de la misma forma con que juega con las leyes de la geometría. Asimismo, prevé un "microcosmos de base de datos" en el que el niño pueda aprender sobre el proceso de la información. Algunas de las ideas de Papert sobre el microcosmos físico ya se han incorporado en el LOGO del Mac. Las simulaciones, los programas CAL y las bases de datos pueden volverse más amables y más potentes en la medida en que programas como el Macintosh Filevision se abran paso en las escuelas.

Sin embargo, las áreas más interesantes de la informática educativa giran alrededor de la introducción de tecnologías completamente nuevas, en particular la CD-ROM, o memoria de lectura solamente en disco compacto, y el video interactivo (IV). La posibilidad de disponer de vastas cantidades de almacenamiento de datos a un precio moderado permite que el software educativo se desmarque del enfoque "lineal" típico de programas desarrollados en sistemas pequeños y poco accesibles al usuario. El grado de interacción de un programa CAL de 48 K es extremadamente limitado y un software de tal clase impone necesariamente restricciones en el proceso del aprendizaje. Con el IV, el programador puede permitir al estudiante muchísima más flexibilidad para enfocar un tema, haciendo uso del mayor espacio de memoria disponible para permitir que durante la ejecución del programa se tomen direcciones diferentes.

Recientemente, un equipo de trabajo del Council for Educational Technology señaló cuatro áreas en las que se podría aplicar el IV en las escuelas. Los maestros lo podrían emplear como un medio auxiliar para la enseñanza, proporcionando programas de instrucción y referencias audiovisuales. Se podría acceder de manera expedita a las imágenes y las secuencias. Grupos reducidos de niños podrían utilizarlo como "profesor", analizando el material y respondiendo a preguntas. A nivel individual, se podría aplicar en simulaciones totalmente interactivas, programas CAL sofisticados y aprendizaje de



La eficacia del video
El uso de micros por departamentos en las escuelas británicas refleja claramente una tendencia muy fuerte hacia las aplicaciones científicas. La introducción de los sistemas de video interactivo y el desarrollo de software IV podrían hacer mucho por reorientar este equilibrio. La enseñanza de historia, por ejemplo, podría experimentar un cambio radical mediante el uso de complejas simulaciones, factibles en función del enorme potencial de almacenamiento de datos de la tecnología del videodisco

Proyecto de envergadura

La enorme capacidad de almacenamiento del videodisco abre posibilidades para aplicaciones completamente nuevas. La BBC, en colaboración con Philips, Thorn-EMI y el Microcomputers in Education Project, está desarrollando a nivel nacional un proyecto para recopilar datos sobre la Gran Bretaña contemporánea. Las escuelas de todo el país reunirán datos relacionados con su entorno inmediato, todos los cuales se compaginarán para formar una "base de datos del pueblo" que contenga detalles sobre el empleo de la tierra, distracciones, servicios, etc. Además, los recursos nacionales de datos (p. ej., el censo y los informes sobre transportes) se combinarán en una inmensa "base de datos nacional", y ambas bases de datos residirán en un único disco láser. El proyecto constituirá un notable documento de la sociedad contemporánea y, gracias a la tecnología del disco láser, contendrá no sólo texto, sino también mapas, diagramas, cuadros e incluso secuencias de imágenes animadas

final abierto. Por último, se podría utilizar para acceder a datos, como una biblioteca de material de referencia audiovisual.

Aunque es necesario un gran trabajo para indexar y acceder a la información y desarrollar software para IV, el sistema ya se está introduciendo en muchas áreas diferentes. La Open University (Universidad Abierta) británica lo ha utilizado en escuelas de verano donde, debido al corto tiempo disponible, los estudiantes de ingeniería tenían dificultades para estudiar el efecto de, por ejemplo, la fatiga en diversos materiales. Se ha escrito un programa IV que emplea fotogramas fijos, animación y secuencias de video. Toma la forma de una batalla en la sala de tribunal entre los fabricantes de ositos de felpa y los proveedores de las arandelas que se utilizan para asegurar los ojos de los ositos. El estudiante debe realizar una serie de experimentos, que conformarán la base de la evidencia utilizada en el tribunal. El programa interactivo se controla mediante un gran programa CAL que "analiza" las posibles causas de defectos de polímero en la substancia que se utiliza para los ojos de los ositos. El estudiante lleva a cabo varios experimentos e interpreta los resultados, que finalmente se presentan en una secuencia dramatizada.

No más escuelas

La transferencia de técnicas de software existentes (simulaciones, CAL, bases de datos, etc.) en sistemas de CD-ROM es apasionante, pero los efectos a largo plazo de la nueva tecnología tienen un alcance mayor. "Yo no creo —afirma Seymour Papert— que de aquí a 20 años existan las escuelas. Es imposible que el concepto de un maestro frente a una clase que toma notas pueda existir en la edad de los ordenadores. Si se repasan una por una todas las cosas que suceden en la escuela, éstas simplemente no serán necesarias, exceptuando aquellas que se dan de forma deficiente, como que los niños hablen entre sí y lleguen a conocerse mutuamente... Existirá algún lugar donde los niños se encuentren y aprendan y se desarrollen. Se llame escuela o no, carece de relevancia. Lo que sí es importante es que la escuela que conocemos en la actualidad ya no existirá."

Sus puntos de vista también los comparte, llevándolos aún más lejos, el profesor Tom Stonier: "El ordenador iniciará el cambio desde la educación basada en la escuela a la educación basada en el hogar. El papel del maestro cambiará, pasando de base de datos andante a consejero de conocimientos."

Las posibles implicaciones de este cambio en la importancia otorgada, de una estructura educativa basada en el sistema a una centrada alrededor del estudiante como individuo, son enormes y no están limitadas de ninguna manera al proceso de aprendizaje. El profesor Stonier es coautor de un libro titulado *Children, computers and communications* (Niños, ordenadores y comunicaciones), en el que predice la educación a través de un sistema de "abuela electrónica".

De acuerdo a Stonier, a medida que la educación pase de la escuela al hogar y las mejoras en el campo de la salud y la medicina incrementen el promedio de vida del hombre, los ciudadanos adultos se irán comprometiendo más en la educación, como en realidad lo hacen ya en muchas de las sociedades llamadas "primitivas". Stonier piensa que los ancianos, uno de nuestros sectores sociales más postergados, están idealmente cualificados para asumir la responsabilidad de una sección del sistema educativo. En Estados Unidos existe ya un ejemplo de cómo podría desarrollarse el sistema. En Portage (Wisconsin) un maestro visita a las familias durante medio día una vez por semana. En esta visita, el maestro evalúa los progresos hechos por el niño durante la semana anterior y decide, en colaboración con los padres, en qué dirección se encaminará el trabajo de la siguiente semana.

Aunque el sistema de Portage se centra en la educación de niños incapacitados, existen todas las posibilidades de que se amplíe a otros estudiantes. En Gran Bretaña es perfectamente legal educar a los hijos en casa si se puede convencer a la autoridad local en materia de educación de que uno es capaz de hacerlo. El número de familias que están optando por salir del sistema es cada vez mayor, y en la actualidad son más de 2 000 las que han "desescolarizado" a sus hijos. El acceso a inmensas bases de datos públicas, los potentes micros con grandes bases de datos de CD-ROM y el software interactivo les ofrecerán a los padres una eficacísima herramienta para educar a sus hijos en el propio hogar.

A medida que aumentan la complejidad y el alcance del software educativo, el papel del maestro se vuelve cada vez menos fácil de definir. Stonier predice que los estudiantes llegarán a estar mejor informados que sus propios maestros. En la medida en que los maestros cumplan con la función de "base de datos caminante", el conocimiento que adquiera un niño se verá severamente limitado por la comprensión del maestro. Todos los profesores de informática, e incluso los de la mayoría de otras asignaturas, pueden señalar un niño que los ha superado. Stonier piensa que este patrón se repetirá a través de todo el plan de estudios a medida que se vayan desarrollando bases de datos y software educativo.

BBC Open University Production Centre

Testigo ocular
Las secuencias cinematográficas y la información visual de alta resolución hacen necesario procesar y almacenar inmensas cantidades de datos; por supuesto, ello es imposible en los sistemas controlados por microordenadores, pero actualmente la tecnología del video interactivo (IV) se está imponiendo con rapidez y a un precio asequible. Estos fotogramas pertenecen a *The teddy bear disc* (El disco del osito de felpa) y muestran un sistema de videodisco controlado por ordenador en plena acción, en este caso un programa sobre materiales de ingeniería para alumnos universitarios. Éste se desarrolla en un tribunal industrial y la tecnología IV permite que el estudiante asuma el papel de "testigo pericial", llevando a cabo experimentos y utilizando el sistema para construir un caso legal

La revolución del CD-ROM

Para almacenar los datos de forma fiable, gran parte del espacio de los CD-ROM debe dedicarse a rutinas de verificación de errores e información de formateo. No obstante, un disco CD-ROM ofrece al usuario 552 megabytes de almacenamiento de datos, lo que permite almacenar en un solo disco millares de pantallas de información. Además, a diferencia de los reproductores de discos CD de audio, que necesitan sofisticados convertidores D/A, los reproductores de CD-ROM sólo han de tratar con datos digitales, y esto simplifica el sistema de circuitos del reproductor y reduce el costo unitario. Teniendo en cuenta que los reproductores, en los otros aspectos, son casi idénticos a las unidades de audio, es probable que los precios permanezcan bajos debido a la demanda del consumidor y las técnicas de fabricación estándar. Los discos de CD-ROM se venderán a precios similares a los de los discos compactos de audio si las ventas son aceptables. Tras su introducción se producirá un auténtico cambio en la comercialización de software

Variables valiosas

Analizamos el uso de funciones y ofrecemos varios ejemplos que ilustran la solidez estructural del PASCAL

Así como los procedimientos se pueden definir y llamar por su nombre, también se puede llamar a las funciones, pero sólo formando parte de expresiones, no como sentencias. Mientras que las llamadas a procedimientos invocan la ejecución de subprogramas para llevar a cabo algún proceso, las llamadas a funciones *calculan* un valor. El resultado devuelto puede ser de cualquier tipo simple, real o escalar, o de tipo *puntero* (que aún no hemos tratado). La única diferencia en el encabezamiento, además de utilizar la palabra reservada FUNCTION, es que se debe especificar el identificador de tipo del resultado devuelto. Supongamos que el PASCAL no dispusiera de la función predefinida impar (*odd*); sería fácil definir una propia:

```
FUNCTION Impar (numero:integer);boolean;
  BEGIN
    Impar:=numero MOD 2>0
  END;{Impar}
```

El valor se devuelve en el identificador de función y, por lo tanto, debe serle asignado a él en el cuerpo de la función, como podemos ver. En este sentido, los nombres de funciones son como variables que no se inicializan nunca, pero sus valores se calculan cada vez que aparecen en una expresión. Sólo pueden aparecer en el lado izquierdo de una asignación, de modo que si fuésemos descuidados y programáramos la asignación como:

```
IF numero MOD 2>0 THEN
  Impar:=true
```

(olvidando la necesaria cláusula ELSE Impar:=false), existe en la construcción un camino que deja el resultado indefinido.

Al igual que con los parámetros de procedimientos que hemos visto hasta ahora, el valor del parámetro actual se pasa al identificador entero local, número, desde el punto de "activación". De modo que:

```
WriteLn (Impar(sqr(N DIV 100)))
```

que siempre imprimiría False, realiza cuatro operaciones:

1. se calcula la expresión N DIV 100
2. este valor entero temporal se pasa a srq como un parámetro de valor actual
3. se devuelve su cuadrado y se le pasa a Impar
4. ahora se calcula el resultado booleano y se pasa al procedimiento WriteLn como otro parámetro de valor.

Si cuando se ejecutara la sentencia de arriba, N tuviera el valor 17, ¿cuál sería después su valor? Ésta puede parecer una pregunta ridícula, por supuesto. No hay ninguna razón posible por la cual el valor de N cambie durante la evaluación de cualquier función que implique a N como un parámetro. Los resultados de todas las funciones deben depender únicamente del "estado del mundo" tal como existe. Es decir, el valor devuelto es una "función" (sic) del (de los) valor(es) de sus "argumentos" o parámetros. El mecanismo del PASCAL para pasar sólo el valor de las variables asegura que, aun cuando los parámetros cambien localmente en virtud de alguna función o algún procedimiento, y debido a que en realidad son copias locales, en el punto de activación los parámetros reales no se verían afectados.

Por estos mismos motivos, aunque el PASCAL no lo prohíbe, los datos no deben ser accedidos globalmente. Todos los enlaces de datos entre llamadas a procedimientos y a funciones se deben controlar mediante listas de parámetros, aunque dichos datos se hallen dentro del ámbito del subprograma. La única excepción a esta regla general es para el acceso a constantes globales. Por su propia naturaleza, éstas no pueden ser dañadas en PASCAL, porque es imposible alterar sus valores. Sin embargo, si se pasara como parámetro un valor de constante, se convertiría en una variable local y, por consiguiente, dejaría de ser inexpugnable.

```
FUNCTION Minuscula (caracter:char)          :char;
  {devuelve la minuscula de cualquier argumento
  mayuscula}
  CONST
    offset=32;{ASCII ord ('a')–ord('A')}
  BEGIN
  IF caracter IN['A'..'Z']
    THEN
      Minuscula:=chr(ord(caracter)+offset)
    ELSE
      Minuscula:=caracter
  END;{Minuscula}
```

Cuando definimos procedimientos, en algunas ocasiones es esencial que *alteren* los valores de sus propios parámetros; de lo contrario, no llevarían a cabo la tarea para que fueron diseñados. Un ejemplo obvio es el propio procedimiento read del PASCAL. No sería muy útil que read (N) diera meramente un valor a un entero que fuera local de read y que no se modificara el valor de N. En tales casos necesitamos pasar la dirección de un parámetro variable (en vez de su valor) con el fin de que el procedimiento se pueda referir directamente a él en lugar de a su copia local. Este mecanismo se denomina *paso por dirección* o *por referencia* y normalmente sólo se debe utilizar con procedimientos y no con funciones.

El paso de parámetros variables se consigue simplemente incluyendo la palabra reservada VAR antes del o de los elementos de la lista de parámetros de procedimiento. Se puede considerar que la sintaxis de una lista de parámetros es idéntica a la declaración VAR de un bloque; pero en vez de que VAR aparezca una sola vez como delimitador, sólo aparece antes de un elemento que necesita ser VARiado por el procedimiento. Por ejemplo:

```
PROCEDURE Proceso (VAR contador : ListaContador);
```

Desarrollo de algunas aplicaciones

Para ilustrar esta potentísima técnica, vamos a desarrollar un programa que lea algunos nombres desde el teclado y, relacionada con cada nombre, una cantidad de dinero que nos debe cada una de esas personas. Por razones de simplicidad, utilizaremos sólo una serie por nombre y expresaremos las deudas como cantidades enteras en pesetas. Una vez probado el programa, podríamos añadir más detalles, como direcciones, números de teléfono, etc. Podríamos imprimir la lista de nombres tanto por orden alfabético ascendente como descendente, o quizá, por el orden de las cantidades adeudadas. La estructura natural de los datos será una lista de registros que contengan campos del tipo apropiado cuya adición resulte fácil posteriormente.

Podemos manipular cada registro como un dato individual y utilizar cualquiera de los campos como "clave" para clasificación. Antes que nada, diseñemos la representación completa de los datos:

```
CONST
  LongSerie    = 20;
  LongLista    = 50;
TYPE
  Cardinal     = 0..MaxInt;
  TamañoSerie  = 1..LongSerie;
  serie        = PACKED ARRAY [TamañoSerie]
                                  OF char;
  dato         = RECORD
{otros campos a    nombre :serie;
 conveniencia}     deuda :Cardinal;
                END;{dato}
  limites      =1..LongLista;
  ListaRegistro =ARRAY [limites] OF dato;
```

Aquí se permiten cincuenta nombres y se da por sentado (de momento) que ningún nombre excederá de 20 caracteres, pero esto lo podemos modificar fácilmente.

Hemos expresado todos los subrangos y estructuras en una parte de definición de TYPE por varios motivos:

1. seguridad: un índice de tipo limites no puede exceder nunca los límites definidos de la matriz, p. ej.
2. localización de errores: en caso de que durante la comprobación se produjera un "error de rango", el mensaje de error ayudaría a localizar el problema.
3. conveniencia y eficacia: las variables locales pueden tener tipos ya existentes, evitando la duplicación y ahorrando memoria.
4. necesidad: el PASCAL insiste en que los "denotadores" de tipo de parámetros deben ser identificadores, no "literales" (0..255, p. ej.).

A continuación, hemos de pensar en unas pocas variables esenciales y en un algoritmo, todo lo cual lo analizaremos en el siguiente capítulo de esta serie. Mientras tanto, pruebe de desarrollar su propio "algoritmo informal" para el programa y confeccione una lista de todos los procedimientos que crea que puedan ser necesarios.

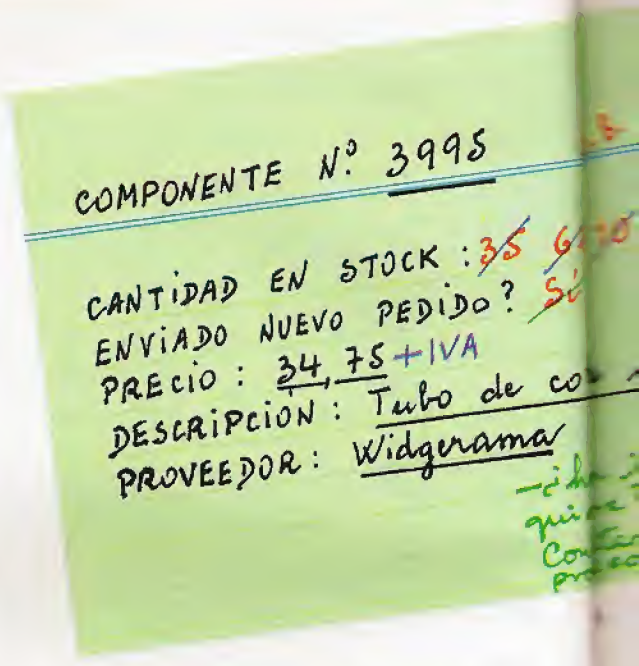
Ventaja positiva

El programa *LectorFichas* manipula números positivos de una forma totalmente segura al leer los datos como caracteres. La conversión al valor numérico equivalente se efectúa restándole el valor original del carácter del dígito al del dígito "0". Esto funciona para todos los juegos de caracteres, porque los caracteres "0"..."9" se definen como contiguos. Observe que el programa posee sólo dos variables globales (en el "nivel 0"). Los datos para procesar los números y la función Valor que lleva a cabo la asociación, los utiliza sólo LeerFicha y, por consiguiente, todo ello está declarado y definido dentro de este procedimiento. Tal como está ahora el programa, sólo tiene un problema: si se entrara un valor mayor que la cardinalidad de los enteros del compilador (MaxInt), la sentencia de asignación del bucle WHILE quedaría colgada. ¿Qué le parece la idea de escribir otro procedimiento LeerInt para hacer frente a números negativos?

```
PROGRAM        LectorFichas     (input, output);

TYPE
        cardinal              = 0..MaxInt;

VAR     PosNUm   : cardinal;
        okay     : boolean;


PROCEDURE  LeerFicha     ( VAR  N     : cardinal;
                           VAR  OK    : boolean);
  CONST
        espacio    = ' ';
  TYPE
        simple     =0.9;
  VAR
        simbolo    :char;
        digitos    :SET OF char;
          {22222222222222222222222}
                {nivel 2}
  FUNCTION  Valor ( digito  : char )     :simple;
        {devuelve el valor numerico de
        un digito en cualquier juego de caracteres}
    BEGIN
    Valor := ord (digito) - ord ('0')

    END;  {Valor}
          {22222222222222222222222}
                {vuelta a nivel 1}
  BEGIN {LeerFicha}


    REPEAT
      read (simbolo)

    UNTIL simbolo>espacio;

    digitos :=['0'..'9'];

    OK := simbolo IN digitos;

    IF OK THEN
      BEGIN
        N := 0;

        WHILE simbolo IN digitos DO
          BEGIN {calcular en base 10}
            N := 10*N-Valor (simbolo);
            read (simbolo)
          END  {se suprime cualquier car. delimitador}
      END      {o 'EoLn' despues del numero}

  END;   {LeerFicha}
         {11111111111111111111111111111111}
BEGIN    {LectorFichas - Nivel 0 Programa Principal}

  page (output);
  WriteLn;
  WriteLn ('Este programa lee enteros positivos');
  WriteLn ('(en la escala 0..',MaxInt:1,')');
  WriteLn ('La terminación del programa se');
  WriteLn ('controla mediante deteccion de error.');
  WriteLn;
  write ('Entre un numero:');
  LeerFicha (PosNum, okay);

  WHILE okay DO
    BEGIN
      WriteLn;
      write ('El numero que entro era:');
      WriteLn (PosNum:1);
      write ('Numero?');
      LeerFicha (PosNum, okay)
    END;

  WriteLn ('--- DETECTADO ERROR ---':40)

END.
```

Entrada doble

Presentamos distintos enfoques de organización de ficheros, desde la base de datos "plana" directa hasta la compleja y poderosa "dBase II"



Todos los registros contenidos en una base de datos "plana" suelen ser totalmente autocontenidos; es decir, cualquier registro dado normalmente retiene toda la información requerida, sin necesidad de referirse a otro registro. Los archivos de fichas tradicionales son planos en este sentido. Podemos pensar, por ejemplo, en una base de datos de biblioteca: cada registro se compone simplemente de los campos TITULO, AUTOR, EDITORIAL e ISBN y no se requiere ninguna referencia cruzada.

Algunas veces, un campo contiene una entrada que puede tener un registro propio. En el archivo de los socios de un club, por ejemplo, un campo HIJOS se puede referir a otros registros del mismo archivo; si los hijos de un socio también son socios del club, ellos tendrán sus propios registros. Pero un campo se puede referir a registros que puedan no tener cabida en la estructura general del archivo de la base de datos. Consideremos una base de datos de inventario de componentes, que contiene los campos NUMERO DE COMPONENTE, PRECIO, CANTIDAD EN STOCK y PROVEEDOR. En este caso, es muy probable que el campo PROVEEDOR también se refiera a otros registros que no responden a la estructura definida para la base de datos de COMPONENTES. Para ilustrar esto veamos, en primer lugar, el archivo SOCIOS DEL CLUB:

NOMBRE DEL SOCIO	Susana Gómez
AÑO DE INSCRIPCION	1979
SUSCRIPCION?	Pagada
OCUPACION	Maestra
SALARIO	89000
HIJOS	Ana Gómez, José Gómez

En este ejemplo, el campo HIJOS contiene dos entradas; ambas podrían tener entradas similares en el archivo SOCIOS DEL CLUB, en el caso de que también fueran socios. Comparémoslo con un registro de la base de datos COMPONENTES:

NUMERO DE COMPONENTE	3995
CANTIDAD EN STOCK	86
PRECIO	34.75
DESCRIPCION	tubo de cobre de 2 m
PROVEEDOR	Widgerama Ltd; Dongle Corp de Taiwan

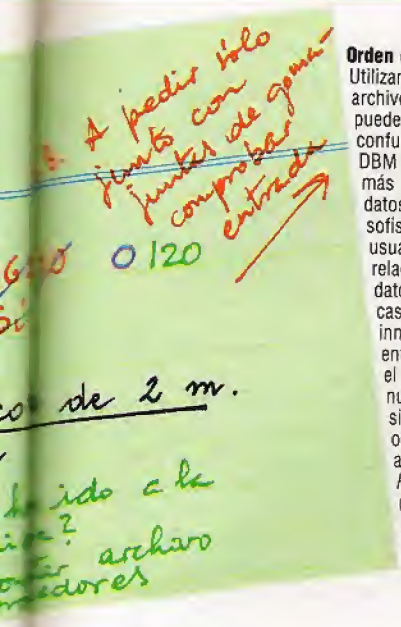
En el campo PROVEEDOR hay dos entradas; ninguna de ellas podría tener registros que se adecuaran a la estructura de la base de datos COMPONENTES. Por consiguiente, un inventario de componentes como éste probablemente tuviera un segundo archivo de base de datos para proveedores, en el cual cada entrada podría ser parecida a la siguiente:

PROVEEDOR	Dongle Corp de Taiwan
DIRECCION	57 Kau Moo Road, Taipei, Taiwan, R.O.C.
TELEFONO	010-886-2-223-4478
SUMINISTRO1	tubo de cobre de 2 m
PRECIO1 ($USA)	34.75
SUMINISTRO2	Pasta para válvulas (lata de ½ litro)
PRECIO2 ($USA)	6.00
SUMINISTRO3	Cera Dazzlebrite (bote pequeño)
PRECIO3 ($USA)	2.30
SUMINISTRO4	Cera Dazzlebrite (bote grande)
PRECIO4 ($USA)	4.00
SUMINISTRO5	—

Es totalmente evidente que la información que necesitamos sobre cada uno de los proveedores es bien diferente, en estructura, de la información que necesitamos sobre los componentes que vende la empresa. La solución consiste en tener dos archivos diferentes: uno para los componentes y otro, separado, para cada uno de los proveedores.

Para implementar con éxito archivos separados pero relacionados entre sí, obviamente es necesario que el DBM sea capaz de manipular más de un archivo a la vez. Los DBM menos sofisticados, como el *Card Box*, del cual ya hemos hablado anteriormente, trabajan sólo con un archivo a la vez, pero los DBM como el *dBase II* pueden usar un archivo como archivo primario (como el de COMPONENTES) y otro archivo (como PROVEEDORES) como archivo secundario. Un buen DBM normalmente permite ejecutar una clasificación en un archivo primario (para extraer registros seleccionados) así como la extracción de registros relevantes (como PROVEEDORES) de archivos relacionados.

El paquete *dBase II* permite utilizar dos archivos relacionados al mismo tiempo, y los asocia median-



Orden de stock
Utilizando su propio sistema de archivo de tarjetas, las entradas pueden ser tan complejas (o confusas) como deseemos. Los DBM deben adoptar un enfoque más estructurado a la entrada de datos, pero los programas sofisticados permiten que el usuario llame a archivos relacionados y, al entrar los datos, se salte ciertos campos en caso de que éstos sean innecesarios en virtud de entradas previas del registro. En el ejemplo, el campo "Enviado nuevo pedido" sólo es relevante si el stock se aproxima a cero o ya lo es. En consecuencia, algunos DBM, como el *Rescue*, permiten que el usuario prepare campos que sean dependientes de campos anteriores o bien que se calculen en función de ellos

te el empleo de campos clave comunes. Dos instrucciones permiten pasar la referencia de uno a otro: SELECT PRIMARY y SELECT SECONDARY (seleccionar primario y seleccionar secundario). Si el archivo primario es COMPONENTES, el secundario, PROVEEDOR, y estuviéramos trabajando en el archivo COMPONENTES, impartiríamos la instrucción USE COMPONENTES. Si después quisiéramos realizar una referencia cruzada al archivo de proveedores, impartiríamos las instrucciones: SELECT SECONDARY y luego, en la siguiente línea, USE PROVEEDOR. Se pueden utilizar todas las instrucciones normales del *dBase II*, que operarán en el archivo que se haya seleccionado más recientemente.

Existen ocasiones en las que el contenido de un campo (como PROVEEDOR en el archivo COMPONENTES mencionado previamente) no requiere las facilidades de registros completamente dependientes (en archivos independientes), de modo que bastarán los campos dependientes dentro del mismo registro. A modo de ejemplo, supongamos que usted lleva una base de datos de la última tecnología en materia de automóviles, almacenando registros de recortes de prensa y referencias a libros que ha visto, relacionados con cosas tan diversas como cuerpos de plástico moldeados a inyección e inyección de combustible controlada electrónicamente, usted podría diseñar un formato como éste:

TEMA: _____
RESUMEN1: _____
RESUMEN2: _____
RESUMEN3: _____
FUENTE: _____ ISBN: _____
TITULO: _____
FECHA: _____
N. PAGINA: _____

Si la fuente de una entrada es una revista, como *Muy Interesante* o *Dirigido por...*, el campo de ISBN será irrelevante, dado que las revistas no poseen ISBN. No obstante, si la fuente de la entrada es un libro llamado *Hacia la comprensión de la informática*, habrá un ISBN. Del mismo modo, si la fuente es un libro o una revista, habrá un número de página. Pero si la fuente fue un programa de televisión, el campo N. PAGINA no será necesario.

Algunos DBM, como el *Rescue*, de Microcomputer Business Systems, permiten que la presencia o la ausencia de cualquier campo en un registro sea calculada o dependa de otros campos previos.

Por consiguiente, si la FUENTE no es un libro, no se solicitará una entrada para el campo ISBN, que no se visualizará (ni en la pantalla ni en las salidas impresas). Si la fuente no es ni un libro ni una revista, el campo N. PAGINA no será necesario y tampoco se visualizará. Un DBM capaz de hacer esto economiza memoria al eliminar los campos innecesarios y ayuda a mejorar la presentación de la información. Será, no obstante, menos versátil que un DBM capaz de relacionar registros de un archivo con registros de otro.

En las bases de datos en las que existe alguna información raíz que permanece constante para todos los registros, y alguna información de ramificación que puede o no ser requerida, tenemos lo que se conoce como *jerarquía de dos niveles*. Como regla general, los DBM de archivos múltiples pueden tratar datos jerárquicos de dos niveles como un conjunto o caso especial de una base de datos de archivos múltiples. Por ejemplo, una referencia FUENTE a un libro podría relacionarse con un archivo LIBROS y una referencia FUENTE a un programa podría relacionarse con un archivo PROGRAMAS.

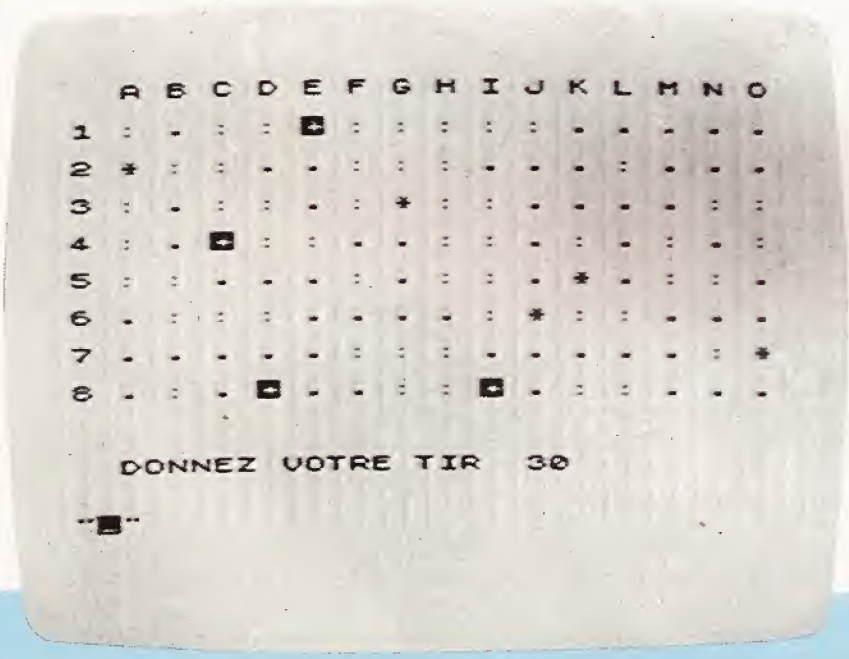
Hay un último punto a tener presente. Un archivo de tarjetas escritas a mano diseñado por usted mismo puede ser tan complejo como desee. Pero habrá serias limitaciones entre los diferentes DBM disponibles comercialmente en cuanto a lo que puedan y no puedan manipular estructuralmente. De modo que antes de adquirir un DBM para su micro, piense qué es exactamente lo que desea que haga y examine con atención las especificaciones para el producto que esté considerando.

Bases de datos al detalle

NOMBRE	FABRICANTE	MAQUINA	FORMATO		OBSERVACIONES
Betabase	Clares	BBC	disco		Long. máx. registro 2048. Campos máx. 200. Tamaño máx. arch. 99 K (40T), 199 K (80T)
Database	Gemini Marketing	Spectrum	cassette		Sistema índice de tarjetas. Regs. definidos por usuario.
DFM Database	Dialog	C64	disco		15 campos/registro. Campo alfanumérico de 36 caracteres. Campo numérico de 9 cifras.
Practifile	Computer Software Associates	C64	disco		3800 reg/archivo. Reacomoda datos para lista de correspondencia. Entrada remota.
MicroPen	AMSoft	Amstrad	disco		Contiene procesador de textos y hoja electrónica incorporados. Long. máx. registro 1024. Núm. máx. registros por archivo:32750.

Batalla naval

Este juego le presenta batalla a su ZX81. Tal vez le sea útil ampliar la memoria a 16 K



Al iniciarse el programa, aparecen las casillas en la pantalla, y el ordenador le solicita las posiciones de sus navíos. A continuación la máquina elige las posiciones de sus propios barcos. Cumplida esta operación, el programa entra en un bucle solicitando vuelta a vuelta las coordenadas de tiro de cada uno de los adversarios. Sus navíos se visualizan en la pantalla bajo la forma de un asterisco, y los barcos hundidos, bajo la de un asterisco invertido. Los sectores alcanzados están representados por ":", y los aún intactos, por ".". El juego finaliza cuando uno de los adversarios ha perdido todos sus barcos. A medida que el juego avanza, el ordenador trabaja de manera más lenta. Es muy importante seguir correctamente el listado al digitar. Para modificar el número de navíos basta cambiar el número 7 en las líneas 250, 330, 401, 460 y 552. Una partida dura entre 25 y 40 minutos.

```
 10 REM BATALLA NAVAL
 20 REM SALTAR SUBPROGRAMAS
 30 GOTO 100
 40 LET X=((((CODE A$(1))-29)+2
    )-1)*2
 50 LET Y=((((CODE A$(2))-38)+2
    )-1)*2)
 60 RETURN
 70 LET T=PEEK (PEEK 16398+256*
    PEEK 16399)
 80 RETURN
100 REM FIJACION
110 PRINT ".";
120 FOR I=38 TO 52
130 PRINT " ";CHR$ (I);
140 NEXT I
150 PRINT
160 PRINT
170 FOR I=29 TO 36
180 PRINT CHR$(I);" ";
185 FOR J=1 TO 15
190 PRINT ", ";
200 NEXT J
204 PRINT
205 PRINT
210 NEXT I
211 RAND
220 PRINT AT 20,0;"COLOQUE SUS
    BARCOS."
230 FOR K=1 TO 60
240 NEXT K
250 FOR I=1 TO 7
260 PRINT AT 20,0;"COORDENADAS
    DEL BARCO ";I
270 INPUT A$
275 IF LEN A$<>2 THEN GOTO 270
280 IF CODE A$(1) <29 OR CODE A$
    (1)>36 OR CODE A$(2) <38 OR CODE
    A$(2) >52 THEN GOTO 270
290 GOSUB 40
300 PRINT AT X,Y;
302 GOSUB 70
304 IF T=CODE:"*" THEN GOTO 270
 05 PRINT "*"
 10 NEXT I
 20 PRINT AT 20,0;"ESCOGÍ MIS
    POSICIONES..."
330 DIM S$(7,2)
340 FOR I=1 TO 7
350 LET A$=CHR$ (INT (RND*8)+29
    )+CHR$ (INT (RND*14)+38)
360 GOSUB 40
365 PRINT AT X,Y;
370 GOSUB 70
380 IF T=CODE "*" THEN GOTO 350
390 LET S$(I)=A$
400 NEXT I
401 FOR J=1 TO 7
402 FOR K=1 TO 7
403 IF J=K THEN GOTO 405
404 IF S$(J)=S$(K) THEN GOTO 340
405 NEXT K
406 NEXT J
410 LET MS=7
420 LET YS=7
421 LET N=0
425 LET N=N+1
430 PRINT AT 20,0;" DE SU TIRO
    ";N;"
440 INPUT A$
445 IF LEN A$<>2 THEN GOTO 440
450 IF CODE A$(1) <29 OR CODE A$
    (1) >36 OR CODE A$(2) <38 OR CODE
    A$(2) >52 THEN GOTO 440
460 FOR I=1 TO 7
470 IF A$=S$(I) THEN GOTO 700
480 NEXT I
490 GOSUB 40
500 PRINT AT X,Y;
510 GOSUB 70
520 IF T<>CODE "." THEN GOTO 440
530 PRINT ":";AT 20,0;A$(1);"
    ,A$(2);"      RATE.
535 FOR I=1 TO 50
536 NEXT I
540 PRINT AT 20,0;"YO TIRO  ";
    N;"
550 LET A$=CHR$ (INT (RND*8)+29
    )+CHR$ (INT (RND*14)+38)
552 FOR H=1 TO 7
553 IF A$=S$(H) THEN GOTO 550
554 NEXT H
560 GOSUB 40
570 PRINT AT X,Y;
580 GOSUB 70
590 IF T=CODE ":" OR T=CODE "■"
    THEN GOTO 550
600 IF T=CODE "*" THEN GOTO 650
610 PRINT ":";AT 20,0;A$(1);"
    A$(2);"      RATE.
620 FOR I=1 TO 50
630 NEXT I
640 GOTO 425
650 PRINT AT X,Y;"■"
651 PRINT AT 20,0;"HUNDIDO.QUEDAN";
660 LET YS=YS-1
670 PRINT YS
675 IF YS=0 THEN GOTO 740
680 FOR I=1 TO 100
690 NEXT I
695 GOTO 425
700 GOSUB 40
701 PRINT AT X,Y;"■"
705 PRINT AT 20,0;"HUNDIDO.QUEDAN";
710 LET MS=MS-1
720 PRINT MS
725 IF MS=0 THEN GOTO 740
730 GOTO 675
740 IF YS=0 THEN LET W$="HE"
750 IF YS=0 THEN LET L$="HA"
760 IF MS=0 THEN LET W$="HA"
770 IF MS=0 THEN LET L$="HE"
780 FOR I=1 TO 20
800 NEXT I
810 CLS
820 PRINT W$;" GANADO."
830 PRINT
840 PRINT L$;" PERDIDO."
850 STOP
```

Seis robots

Esta vez examinaremos un "kit" que permite construir seis robots distintos con el mismo conjunto de piezas

A pesar de que muchas personas relacionadas con la industria del ordenador personal han aventurado durante algún tiempo que la próxima área de gran crecimiento dentro de la industria sería la robótica personal, las predicciones aún no son una realidad. Los motivos son fáciles de detectar. A pesar del hecho de que durante el último año han salido al mercado numerosos "robots personales", han adolecido de algunos problemas graves que han impedido que el público se convenza de que una introducción a la robótica es una actividad interesante y valiosa.

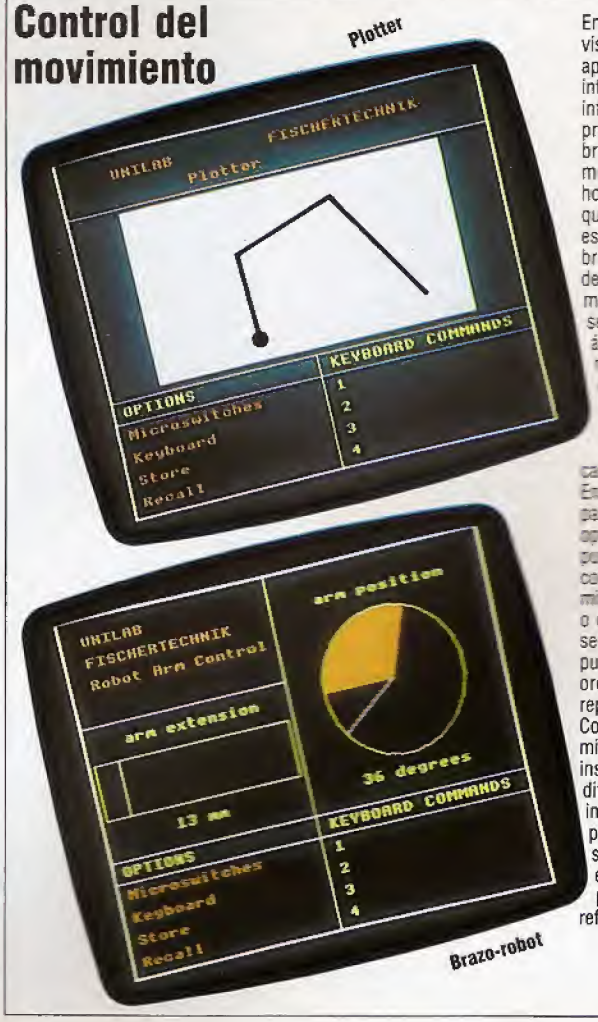
Por otra parte, están los robots de construcción sencilla, como los Movits, que los puede montar en pocas horas cualquier persona que no tenga ningún conocimiento de robótica e incluso ni siquiera de electrónica. En este caso el problema es que estos dispositivos tienen unas aplicaciones muy limitadas y probablemente el usuario pierda enseguida el interés por las máquinas. Por otra parte, existen robots más caros que por lo general exigen un amplio conocimiento de robótica y electrónica para poder manejarlos. Aun así, muchos de estos robots más avanzados poseen sólo una única finalidad. Nuevamente, esto significa que, tras haber explorado todas las rutas, el robot quedará olvidado acumulando polvo en un estante.

Por lo tanto, parece obvio que lo que se requiere es un robot que sea fácil de construir y posea numerosas funciones diferentes que puedan ser exploradas por el usuario. Esto es lo que se pretende con el Fischertechnik Robotics Kit. La idea que ha servido para crear este *kit* es bastante simple. Ya existen desde hace muchos años numerosos juegos para armar electrónicos, dirigidos a los niños, que les permiten realizar circuitos sencillos para construir, por ejemplo, una radio básica. Este mismo *kit* se puede luego desmontar y volver a componer para crear un sencillo dispositivo temporizador electrónico o una alarma antirrobo. Simultáneamente, generaciones de niños han crecido con juegos de construcción Lego que se pueden utilizar una y otra vez para construir una enorme cantidad de modelos diferentes a partir de los mismos bloques plásticos. Al diseñar sus *kits*, Fischertechnik ha adoptado estas dos ideas y las ha combinado con la moderna tecnología del ordenador para desarrollar un producto que bien podría convertirse en el punto de penetración popular para la robótica que tanta gente ha estado prediciendo.

El *kit* Robotics permite que el usuario construya varios diferentes dispositivos robot que se pueden hacer funcionar desde un micro personal. Estos robots van desde un brazo-robot y una máquina que clasifica por orden elementos de longitudes diversas, hasta un plotter y un dispositivo para entrada de gráficos. Una vez que el usuario ha experimentado con un robot determinado, lo puede desarmar y reconstruir para hacer otro. Para poder operar los robots desde un ordenador, se requiere una interface para convertir las señales digitales del ordenador en aquellas que se puedan utilizar para operar los motores eléctricos del *kit*.

Los componentes del juego de construcción constan de varias piezas de plástico y se pueden unir entre sí (al estilo de un juego Lego) para formar diversos modelos. Estas piezas tienen forma de bloques que se pueden unir entre sí para formar las piezas más largas de los brazos, o utilizar para sostener las unidades de control. Otras piezas para construir las unidades mecánicas son ruedas dentadas y tornillos giratorios para hacer girar los dientes. También hay un gran tablero plástico que mide 260×187 mm, que se utiliza ya sea como base para el propio robot o bien, en caso de que se incorpore el plotter o la tablilla para gráficos, como la base para el papel. El sistema de potencia y control eléc-

Control del movimiento



En las fotografías vemos las visualizaciones en pantalla de la apertura del software para la interface Unilab. La fotografía inferior pertenece a un programa de control para el brazo-robot. El diagrama muestra el movimiento horizontal del brazo. Se supone que la posición inicial del robot es cero. A medida que el brazo-robot se mueve hacia la derecha (con la línea moviéndose en la pantalla en sentido horario), aumentará el ángulo en grados. Del mismo modo, cuando se mueva la extensión del brazo, la línea dentro de la barra y la cantidad de milímetros desplazados también cambiarán consecuentemente. En la parte inferior de la pantalla están las diversas opciones de control que se pueden aplicar. Estas se controlan desde los microinterruptores de la placa o desde el teclado. Las secuencias de movimientos se pueden almacenar en el ordenador y recuperarlas y reproducirlas posteriormente. Con el plotter se utiliza el mismo sistema de instrucciones, siendo la única diferencia la adición de instrucciones en el dispositivo para levantar y bajar el lápiz sobre el papel. A medida que el plotter se desplaza por el papel, sus movimientos se reflejan en la pantalla

trico también se debe ensamblar a partir de componentes separados. El juego incluye diversos conectores macho y hembra, ocho dispositivos de conmutación y un par de motores eléctricos. Se proporciona, asimismo, un metro de cable plano de 20 vías y un pequeño trozo extra de cable para realizar "parches".

En general, las piezas son de construcción sólida y tienen aspecto de poder ofrecer muchos años de servicio. La única excepción en este sentido parecen ser los potenciómetros. La placa base de uno de ellos se rompió durante la construcción y, si bien esto no incidió en el funcionamiento del brazo (que vemos en la fotografía) porque la placa base se podía colocar a presión en su sitio, sí plantea una duda sobre su fiabilidad a largo plazo. Otro punto en relación a los potenciómetros es que, a diferencia del resto del cableado, que se puede atornillar en conectores, no hay ningún método para fijarles los cables pelados como no sea envolviéndolos alrededor de los conectores. En este caso es aconsejable que el aficionado que pretenda darle un gran uso al *kit*, suelde las junturas de modo permanente para evitar cortocircuitos que podrían dañar los potenciómetros.

Instalación de los componentes

Construir los robots es bastante sencillo y las piezas en su mayoría son suficientemente grandes como para que las puedan manipular los dedos menos diestros. El principal problema que se plantea en la construcción es el manual, que probablemente sea el punto más débil de todo el sistema. Si bien posee algunas partes estimables (p. ej., las conexiones del circuito son fáciles de seguir), se sustenta en gran medida en fotografías de poca calidad de los modelos parcialmente construidos. Junto con vistas esquemáticas del robot en construcción se muestran fotografías de las piezas que se requieren. El problema es que estas fotografías no ofrecen grandes detalles y carecen de comentarios, por lo cual a menudo los usuarios se encontrarán escudriñando una fotografía para tratar de descubrir dónde se supone que debe ir una pieza determinada. Además, como las fotos muestran al robot en una sola dirección, con frecuencia uno tiene que adivinar dónde está instalada una pieza cuando su posición no aparece a la vista.

No obstante, tal vez lo más grave sea que las fotos son en blanco y negro. Esto significa que cuando finalmente uno llega a la parte final, que es instalar el cableado del circuito en el robot, es casi imposible dilucidar el procedimiento de cableado correcto. Esto es particularmente extraño si se considera que el circuito depende en gran medida de la codificación por colores del cable plano. Los escuetos comentarios que se ofrecen en los diagramas de cableado tampoco son de gran ayuda, diciendo, por ejemplo, que los cables de E3 a E8 pertenecen a una sección determinada.

Dicho esto, hemos de señalar que el robot que presentamos fue construido por alguien que, sin tener ningún conocimiento de robótica ni de electrónica, consiguió poner la máquina en funcionamiento, si bien hubiera sido mejor que Fischertechnik no hubiera confiado en el sentido común de la gente y hubiera proporcionado simplemente un mejor manual de construcción.

Una vez completado el robot, la siguiente tarea consiste en conectarlo a la interface. Aquí, felizmente, el procedimiento es mucho más simple. La placa de interface se compone de varios conectores divididos por secciones que retienen los cables del cable plano. La interface que vemos en la fotografía es para el BBC Micro, y las placas de interface para otros ordenadores probablemente no ofrezcan diferencias sustanciales. En el rincón derecho de la placa hay entradas de interruptores. La lógica de las entradas de interruptores está gobernada por el registro de datos y el registro de dirección de datos del BBC Micro y es similar en operación a la caja buffer que diseñamos en nuestro proyecto del apar-

Electroimán
La barra metálica de encima del electroimán se coloca en el efector final de un brazo-robot. El imán se puede activar y desactivar permitiendo que el robot levante las pequeñas placas metálicas incluidas en el *kit*

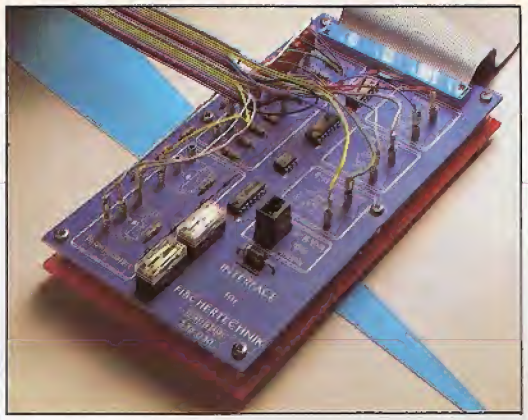
Potenciómetro
Se utiliza para realimentar con datos al ordenador, de manera que sea posible evaluar la posición correcta del robot

Motor
Se halla contenido en una carcasa plástica. La conexión al motor se realiza mediante dos conectores. El tornillo giratorio se utiliza para activar mecanismos de tornillo que activan los robots

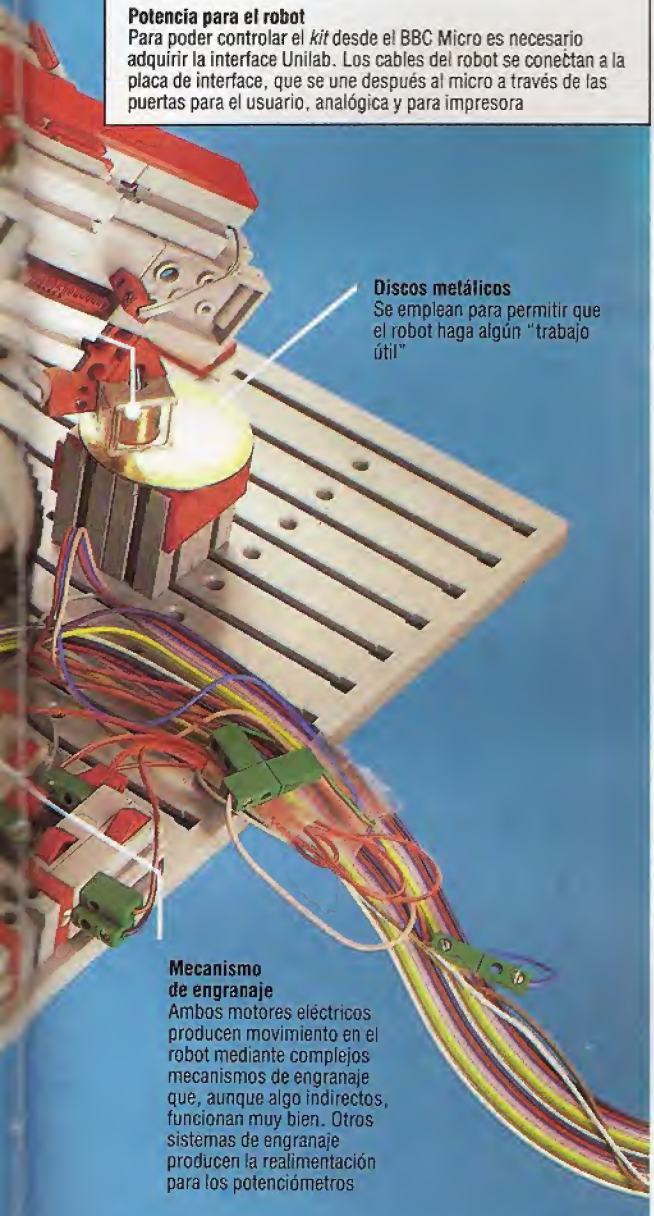
Luces
Siempre que se halle comprometido alguno de los tres sistemas principales de control (imán, motor horizontal o motor vertical), se encenderá una de estas bombillas para informar al usuario cuál es el control que se está utilizando

Interruptor
Al pulsar una combinación de los ocho interruptores se genera un número en el registro de la puerta para el usuario, mediante el cual se puede utilizar el registro de dirección de datos para controlar al robot directamente

Ian McKinnell



Potencia para el robot
Para poder controlar el *kit* desde el BBC Micro es necesario adquirir la interface Unilab. Los cables del robot se conectan a la placa de interface, que se une después al micro a través de las puertas para el usuario, analógica y para impresora



Discos metálicos
Se emplean para permitir que el robot haga algún "trabajo útil"

Mecanismo de engranaje
Ambos motores eléctricos producen movimiento en el robot mediante complejos mecanismos de engranaje que, aunque algo indirectos, funcionan muy bien. Otros sistemas de engranaje producen la realimentación para los potenciómetros

Chris Stevens

tado *Bricolaje*. Debajo de ella hay un par de potenciómetros, que permiten que el ordenador controle la potencia que está llegando a los interruptores y las bombillas. El control de los potenciómetros se mide a través de la puerta analógica.

En el otro lado de la placa de interface hay conexiones para controlar los motores y el electroimán (este último lo utiliza el brazo-robot para asir objetos metálicos). Por último, hay un conector para fuente de alimentación de entre 6 y 8 V que alimenta todo el sistema. La interface se conecta al BBC Micro a través de tres puertas separadas (la puerta para el usuario, la puerta analógica y la puerta para impresora) que se emplean para controlar los motores y el electroimán. Debido a que el sistema está diseñado para usuarios sin experiencia, la placa de interface se ha construido para impedir la posibilidad de un cableado incorrecto que pudiera permitir que se les aplicaran a las interfaces del ordenador voltajes que dañaran la máquina.

El manual que se proporciona para la interface (escrito por Unilab, el fabricante del producto) es muy superior al manual de Fischertechnik Robotics. Hay una detallada explicación sobre cómo preparar la interface y cómo ejecutar el software que se suministra con el sistema. Asimismo, hay un breve resumen acerca de las direcciones de control utilizadas para controlar el sistema, permitiendo, por consiguiente, controlar los robots desde BASIC. Sin embargo, el diagrama del cableado parece estar en desacuerdo con el circuito Fischertechnik. Aunque con la interface se proporciona una nota que hace mención a los cables planos diferentes, ésta no parece esclarecer todos los problemas y, nuevamente, uno queda librado a su talento personal para cablear el circuito en la forma correcta.

El lenguaje PROF

Con la interface se suministra una cassette que contiene tres programas para ayudar a controlar los robots. Los dos primeros son programas especializados que controlan el brazo y el plotter, permitiendo controlar dispositivos externos ya sea desde el teclado o bien desde los interruptores de control del propio robot. El tercer programa es PROF, un sistema operativo generalizado a través del cual uno puede dirigir directamente los robots. Esencialmente, es una ampliación de BASIC con la adición de unas pocas instrucciones. Muchas de las palabras extras introducidas en PROF corresponden a nombres de variables tales como Motor y Magnet para la dirección de control, mientras que otras se incluyen para ayudar a editar secuencias de acciones a ejecutar por los robots.

Aunque el PROF es bastante amplio y admite bucles condicionados y programación estructurada, se ciñe mayormente al control de salida. Por consiguiente, el panel para gráficos, un dispositivo de entrada, no está soportado por el PROF, porque el lenguaje no hace ninguna provisión para el almacenamiento de información desde dispositivos de entrada. Ello requiere que uno tenga que escribir sus programas en BASIC.

A pesar de los problemas de la documentación, el *kit* Robotics y la interface constituyen, en conjunto, un sistema muy valioso. Los robots trabajan sorprendentemente bien y se puede construir un modelo operativo desde cero en unas pocas horas. Y el usuario no se ve limitado a tan sólo los seis diseños que se proporcionan. Una vez alcanzada cierta familiaridad con el sistema, la cantidad de proyectos que se pueden construir es inmensa. Por su precio no se trata, ciertamente, de un juguete que pueda adquirirse con calderilla, y, una vez que se supriman los errores existentes en la documentación, el juego para armar Fischertechniks Robotics y la interface serán una introducción ideal para quien tenga un interés general por el campo de la robótica.

FISCHERTECHNIK ROBOTICS KIT

MOVIMIENTO
La potencia de los robots proviene de uno o dos motores eléctricos CD

CONTROL
Ya sea desde el teclado del ordenador o bien a través de los ocho interruptores del propio robot

FUENTE DE ALIMENTACION
Una sola pila PJ996/4R25 de 6 V

PROVEEDORES
Commotion, Computer Operated Motion, 241 Green Street, Enfield EN3 7SJ, Middlesex, Gran Bretaña

Superar las adversidades

Comenzamos la implementación de seis eventos mayores en nuestro programa de simulación de un viaje comercial al Nuevo Mundo durante el s. XVI

En este capítulo comenzaremos a programar las contingencias mayores. Habrá seis de estos eventos, cada uno de los cuales se puede producir sólo una vez durante la travesía. Se los denomina *mayores* porque sus efectos sobre el viaje son más complejos y de mayor alcance que los producidos por los eventos menores. Al comienzo de cada semana se seleccionará al azar uno de los seis acontecimientos, pero si el evento seleccionado ya se hubiera producido durante el viaje, esa semana no tendrá lugar ningún otro acontecimiento mayor.

La línea 49 DIMensiona una matriz, M(), que se utilizará para señalar si alguna de las seis contingencias mayores se ha producido ya. El programa se envía a la subrutina 6500, que genera un acontecimiento desde el bucle principal del programa mediante un GOSUB a la línea 870. La línea 6508 selecciona un número al azar entre 1 y 10 y, dado que sólo hay seis posibles eventos, hay un 60 % de probabilidades de que se seleccione un acontecimiento; de generarse 7, 8, 9 o 10, no se producirá ninguno mayor. Las probabilidades se pueden alterar, si así se desea, regulando la gama del factor aleatorio de esta línea.

Por ejemplo, al cambiar la línea 6508 por

```
X=INT(RND(1)*8)+1
```

se incrementan las probabilidades de seleccionar un acontecimiento mayor a 6 entre 8, es decir, a un 75 %. No obstante, es necesario comprender que si se selecciona un evento mayor y se establece el indicador correspondiente en la matriz M(), entonces la probabilidad de seleccionar un segundo acontecimiento mayor se reduce a 5 entre 8 (un 62,5 %). Las probabilidades de seleccionar un tercer acontecimiento mayor se reduce aún más, a 4 entre 8, o a un 50 %. Por consiguiente, cuanto más eventos se produzcan, menores son las probabilidades de que se produzca otro acontecimiento hasta entonces no seleccionado.

La subrutina de eventos se selecciona utilizando la sentencia ON X GOSUB. Ésta se emplea de la misma forma en que la sentencia ON X GOTO seleccionaba los eventos menores. Si X, el número seleccionado de forma aleatoria, es 1, enviará el programa al primer número de línea de la lista de subrutinas que siga a la sentencia, la subrutina 6530 (el avistamiento de un bote salvavidas). Si X es 2, lo enviará a la segunda dirección de la lista, la subrutina 6700 (la enfermedad mortal). En este capítulo vamos a cubrir solamente las dos primeras contingencias, pero más adelante habremos de añadir a la línea 6510 las direcciones de las cuatro subrutinas restantes.

En la primera subrutina se avista mediante el catalejo un bote salvavidas. En el bote hay cuatro personas y un gran cofre, y el jugador debe decidir si los sube o no a bordo. La desviación añadirá dos días a la duración del viaje y supondrá cuatro bocas más a alimentar, pero si la tripulación se halla por debajo de su fuerza óptima, quizá los brazos extras sean bien recibidos. El programa comprueba primero si el bote salvavidas ya ha sido avistado previamente, examinando el primer elemento de la matriz M() en la línea 6535. Si está establecido en 1, el evento ya se ha producido y retorna al programa principal. Si es 0, la línea 6540 establece M(1) en 1 para impedir que más adelante en el juego el acontecimiento se vuelva a repetir. Como hemos mencionado antes, esta comprobación se utiliza al comienzo de cada una de las contingencias mayores.

Se le pregunta al jugador si se debe recoger a la tripulación del naufragio, en la línea 6576. La línea 6580 comprueba si el primer carácter de la respuesta es S o N, y, si no lo es, retorna a 6568 para volver a formular la pregunta. Si la respuesta es No, el bote salvavidas se aleja flotando y desaparece, y el control retorna al programa principal. Si prevalece la compasión por los náufragos y el jugador decide efectuar el rescate, el programa comprueba si hay espacio suficiente, determinando en primer lugar, en la línea 6625, si la cantidad máxima de tripulantes a bordo está completa. De ser así, en el barco no habrá lugar para los náufragos (lo que se indica mediante CN=16) y el control retorna al programa principal. Para ver si hay lugar para alguno de los supervivientes, la línea 6630 cuenta los espacios vacíos que hay en el barco estableciendo el valor de X en 16 menos el número de tripulantes, CN, y el programa decide si hay lugar para todos los supervivientes en la línea 6630, comprobando si X es mayor que 3. De ser así, son rescatados; pero si X es menor que 3, se le dice al jugador que sólo hay lugar para X personas más, quienes son entonces subidas a bordo.

Los supervivientes se incorporan a la matriz de fortaleza/categoría en el bucle entre las líneas 6638 y 6697. El valor de X se establece inicialmente en 0 y se utiliza para llevar un contador de tripulantes extras. Se efectúa una comprobación a través de la matriz TS(,), para ver si la categoría de tripulante es 0, lo que significa un lugar en la matriz para un tripulante adicional. De hallarse un lugar, el valor de X se incrementa en 1, indicando que ha sido añadido uno de los supervivientes. Cuando X llega a 4, se habrán incorporado todos los supervivientes a la matriz de la tripulación y, por tanto, la línea 6655 establece el valor de T en 16, haciendo que el programa salga del bucle al llegar al NEXT. Si hay espacio para menos de cuatro tripulantes extras, el programa saldrá del bucle cuando la matriz se haya

verificado 16 veces. Los tripulantes extras también se suman al valor de la variable CN, el contador de la tripulación. A cada nuevo tripulante se le asigna un valor al azar para categoría y fortaleza dentro de la matriz de fortaleza/categoría, TS(,); la categoría se selecciona en la escala del 1 al 5 y la fortaleza en la del 50 al 99. Esto significa que la tripulación extra se catalogará como sana o muy sana.

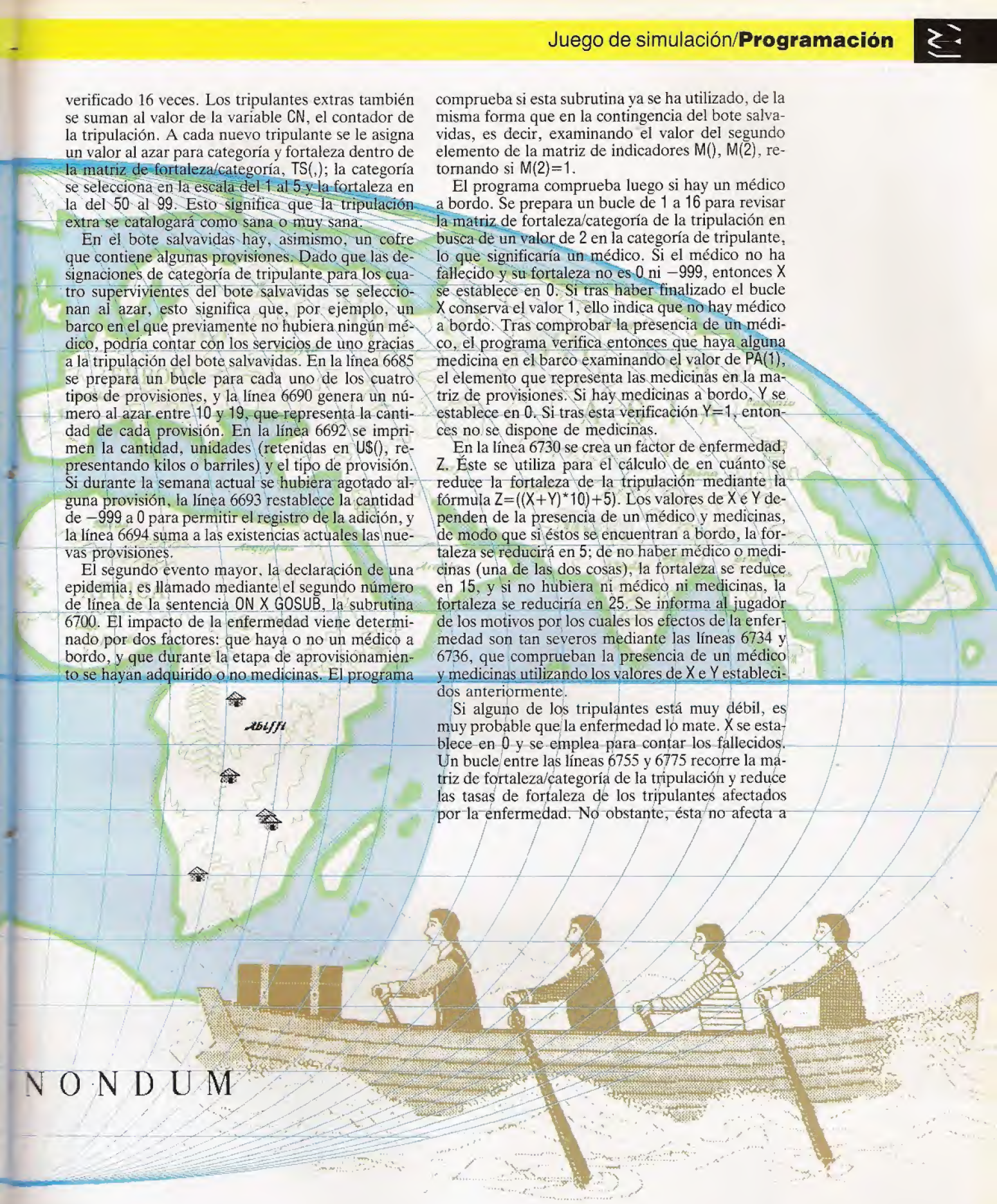
En el bote salvavidas hay, asimismo, un cofre que contiene algunas provisiones. Dado que las designaciones de categoría de tripulante para los cuatro supervivientes del bote salvavidas se seleccionan al azar, esto significa que, por ejemplo, un barco en el que previamente no hubiera ningún médico, podría contar con los servicios de uno gracias a la tripulación del bote salvavidas. En la línea 6685 se prepara un bucle para cada uno de los cuatro tipos de provisiones, y la línea 6690 genera un número al azar entre 10 y 19, que representa la cantidad de cada provisión. En la línea 6692 se imprimen la cantidad, unidades (retenidas en U$(), representando kilos o barriles) y el tipo de provisión. Si durante la semana actual se hubiera agotado alguna provisión, la línea 6693 restablece la cantidad de −999 a 0 para permitir el registro de la adición, y la línea 6694 suma a las existencias actuales las nuevas provisiones.

El segundo evento mayor, la declaración de una epidemia, es llamado mediante el segundo número de línea de la sentencia ON X GOSUB, la subrutina 6700. El impacto de la enfermedad viene determinado por dos factores: que haya o no un médico a bordo, y que durante la etapa de aprovisionamiento se hayan adquirido o no medicinas. El programa comprueba si esta subrutina ya se ha utilizado, de la misma forma que en la contingencia del bote salvavidas, es decir, examinando el valor del segundo elemento de la matriz de indicadores M(), M(2), retornando si M(2)=1.

El programa comprueba luego si hay un médico a bordo. Se prepara un bucle de 1 a 16 para revisar la matriz de fortaleza/categoría de la tripulación en busca de un valor de 2 en la categoría de tripulante, lo que significaría un médico. Si el médico no ha fallecido y su fortaleza no es 0 ni −999, entonces X se establece en 0. Si tras haber finalizado el bucle X conserva el valor 1, ello indica que no hay médico a bordo. Tras comprobar la presencia de un médico, el programa verifica entonces que haya alguna medicina en el barco examinando el valor de PA(1), el elemento que representa las medicinas en la matriz de provisiones. Si hay medicinas a bordo, Y se establece en 0. Si tras esta verificación Y=1, entonces no se dispone de medicinas.

En la línea 6730 se crea un factor de enfermedad, Z. Éste se utiliza para el cálculo de en cuánto se reduce la fortaleza de la tripulación mediante la fórmula Z=((X+Y)*10)+5). Los valores de X e Y dependen de la presencia de un médico y medicinas, de modo que si éstos se encuentran a bordo, la fortaleza se reducirá en 5; de no haber médico o medicinas (una de las dos cosas), la fortaleza se reduce en 15, y si no hubiera ni médico ni medicinas, la fortaleza se reduciría en 25. Se informa al jugador de los motivos por los cuales los efectos de la enfermedad son tan severos mediante las líneas 6734 y 6736, que comprueban la presencia de un médico y medicinas utilizando los valores de X e Y establecidos anteriormente.

Si alguno de los tripulantes está muy débil, es muy probable que la enfermedad lo mate. X se establece en 0 y se emplea para contar los fallecidos. Un bucle entre las líneas 6755 y 6775 recorre la matriz de fortaleza/categoría de la tripulación y reduce las tasas de fortaleza de los tripulantes afectados por la enfermedad. No obstante, ésta no afecta a



todos. La línea 6756 genera un número al azar que, si es menor que 0.3, significa que la fortaleza del tripulante del elemento en curso de la matriz no se ve afectada por la enfermedad. Ello le otorga a cada tripulante alrededor de un 30 % de probabilidades de escapar de la enfermedad (tal vez usted quiera alterar este factor, o, posiblemente, determinarlo al azar). La línea siguiente comprueba si la persona ya está muerta y, por tanto, fuera del alcance de los efectos de la enfermedad. Si el tripulante está vivo, la fortaleza se reduce en Z unidades y se efectúa una comprobación para ver si el tripulante ha muerto a causa de la enfermedad, reduciendo la tasa de fortaleza por debajo de 0. De ser así, se restablece a 999, donde se lo recogerá en el informe de final de semana y el valor de X, la cantidad de fallecidos a causa de la enfermedad, se incrementará en 1. De haber habido medicinas disponibles, lo que se indica mediante Y=0, la línea 6776 reduce la cantidad a la mitad y redondea al frasco entero más próximo. Si durante la epidemia no ha muerto ningún miembro de la tripulación (X=0), entonces el control retorna al programa principal. De lo contrario, se le informa al jugador sobre el número de tripulantes que se vieron mortalmente afectados.

Mód. 8: Dos eventos mayores

Dimensionar matriz bandera

```
49 DIM M(6):REM SEÑALA SI YA SE HAN PRODUCIDO LOS
EVENTOS MAYORES
```

Adición al bucle principal del viaje

```
870 GOSUB 6500:REM IR A EVENTO MAYOR
```

Rutina de selección de eventos mayores

```
6500 REM EVENTOS MAYORES
6508 X=INT(RND(1)*10)+1
6510 ON X GOSUB 6530,6700
6520 RETURN
```

Rutinas de eventos mayores

```
6530 REM BOTE SALVAVIDAS
6535 IF M(1)=1 THEN RETURN
6536 PRINTCHR$(147)
6540 M(1)=1
6550 S$="SE HA AVISTADO UN BOTE SALVAVIDAS*":GOSUB
     9100
6552 S$="NAVEGANDO A LA DERIVA A LO LEJOS*":GOSUB 9100
6554 PRINT:GOSUB 9200
6556 S$="A TRAVES DEL CATALEJO VES*":GOSUB 9100
6558 S$="QUE CONTIENE:*":GOSUB 9100
6560 PRINT:GOSUB 9200
6562 S$="4 PERSONAS*":GOSUB 9100
6563 GOSUB 9200
6564 S$="Y UN GRAN COFRE!*":GOSUB 9100
6566 PRINT:GOSUB 9200
6568 S$="SI ALTERAS TU RECORRIDO*":GOSUB 9100
6570 S$="PARA RECOGERLOS*":GOSUB 9100
6572 S$="TARDARAS DOS DIAS MAS*":GOSUB 9100
6574 PRINT:GOSUB 9200
6576 S$="QUIERES RESCATARLOS (S/N)*":GOSUB 9100
6578 INPUT I$: I$=LEFT$(I$,1)
6580 IF I$<>"S" AND I$<>"N" THEN 6578
6585 IF I$="S" THEN 6600
6588 PRINT:GOSUB 9200
6590 S$="EL BOTE SALVAVIDAS DESAPARECE...*":GOSUB 9100
6592 PRINT:GOSUB 9200
6594 S$=K$:GOSUB 9100
6596 GET I$:IF I$=""THEN 6596
6599 RETURN
6600 PRINT:GOSUB 9200
6610 EW=EW+2/7
6625 IF CN<>16 THEN 6630
6627 S$="NO PUEDES RECOGERLOS*":GOSUB 9100
6628 S$="PORQUE NO TIENES LUGAR EN EL BARCO*":GOSUB
     9100
6629 GOTO 6592
6630 X=16 - CN:IF X>3 THEN 6635
6632 S$="EN EL BARCO SOLO TIENES LUGAR PARA*":GOSUB
     9100
6633 PRINT X;"PERSONAS MAS"
6634 PRINT:GOSUB 9200
6635 S$="RECOGES:*":GOSUB 9100
6638 X=0
6640 FOR T=1 TO 16
6645 IF TS(T,1)<>0 THEN 6679
6650 X=X+1
6655 IF X>4 THEN T=16:GOTO 6679
6660 CN=CN+1
6665 TS(T,1)=INT(RND(1)*5)+1
6668 TS(T,2)=INT(RND(1)*50)+50
6670 PRINT "1  ";C$(TS(T,1))
6679 NEXT
6680 PRINT:GOSUB 9200
6682 S$="EL COFRE CONTIENE:*":GOSUB 9100
6685 FOR T=1 TO 4
6690 X=INT(RND(1)*10)+10
6692 PRINT X;U$(T);"S DE  ";P$(T)
6693 IF PA(T)= -999 THEN PA(T)=0
6694 PA(T)=PA(T)+X
6695 NEXT
6699 GOTO 6592
6700 REM AZOTE DE LA EPIDEMIA
6705 IF M(2)=1 THEN RETURN
6706 PRINTCHR$(147)
6710 M(2)=1
6712 S$="SE DECLARA UNA EPIDEMIA*":GOSUB 9100
6714 PRINT:GOSUB 9200
6716 X=1
6718 FOR T=1 TO 16
6720 IF TS(T,1)=2 AND TS(T,2)<>0 AND TS(T,2)<>-999THEN
     X=0:T=16
6722 NEXT
6724 Y=1
6726 IF OA(1)<>0 AND OA(1)<>-999 THEN Y=0
6730 Z=((X+Y)*10)+5
6732 I$="TIENES  ":IF X=0 AND Y=0 THEN 6740
6734 IF X=1 THEN S$="SIN NINGUN MEDICO*":GOSUB
     9100:I$="NI  "
6736 IF Y=1 THEN S$=I$+"NINGUNA MEDICINA*":GOSUB 9100
6740 S$="MUCHOS DE LOS TRIPULANTES ESTAN
     AFECTADOS*":GOSUB 9100
6745 PRINT:GOSUB 9200
6750 X=0
6755 FOR T=1 TO 16
6756 IF RND(1)< .3 THEN 6775
6760 IF TS(T,2)=0 OR TS(T,2)=-999 THEN 6775
6765 TS(T,2)=TS(T,2)-Z
6770 IF TS(T,2)<1 THEN TS(T,2)=-999:X=X+1
6775 NEXT
6776 IF Y=1 THEN 6780
6777 S$="SE HAN UTILIZADO LA MITAD DE TUS
     MEDICINAS*":GOSUB 9100
6778 OA(1)=INT((OA(1)/2)+.5)
6780 PRINT:GOSUB 9200
6785 IF X=0 THEN 6797
6790 PRINT"Y";X;
6792 S$="TRIPULANTES MURIERON*"
6794 IF X=1 THEN S$="TRIPULANTE MURIO*"
6795 GOSUB 9100
6796 PRINT:GOSUB 9200
6797 S$=K$:GOSUB 9100
6798 GET I$:IF I$="" THEN 6798
6799 RETURN
```

Complementos al BASIC

Spectrum:

Introduzca las siguientes modificaciones:

```
6510 IF X=1 THEN GO SUB 6530
6511 IF X=2 THEN GO SUB 6700
6536 CLS
6578 INPUT I$:LET I$=I$(1 TO 1)
6596 LET I$=INKEY$:IF I$="" THEN GO TO 6596
6706 CLS
6798 LET I$=INKEY$:IF I$="" THEN GO TO 6798
```

BBC Micro

Introduzca las siguientes modificaciones:

```
6536 CLS
6596 I$=GET$
6706 CLS
6798 I$=GET$
```

Por caminos diferentes

En esta ocasión crearemos el software necesario para controlar nuestro brazo-robot desde un Commodore 64, que será distinto del que se precise para hacerlo desde un BBC Micro

El sistema operativo del Commodore 64 genera interrupciones periódicamente para realizar operaciones de administración doméstica, como puede ser la exploración del teclado. Estas interrupciones se producen regularmente a intervalos de un sesentavo de segundo. El programa en código máquina Commodore intercepta el vector de interrupción para llevar a cabo primero su propia rutina antes de ceder el control a la rutina de interrupciones normal. Este tipo de programa suele denominarse *cuña*.

La primera sección del código máquina se ocupa, por lo tanto, de intercambiar el vector IRQ normal con el vector para nuestra rutina.

Una vez que nuestro vector ha reemplazado al vector IRQ normal, nuestro manejador de eventos se implementará cada vez que se produzca una interrupción IRQ. (El código manejador de eventos es, en gran parte, el mismo que el programa de control para servomotores múltiples que hemos ofrecido anteriormente.) En esencia, este código envía impulsos de corriente a lo largo de las líneas de datos desde la puerta para el usuario hacia los motores. La duración de cada impulso le informa al motor correspondiente el ángulo que debe adoptar. La rutina en código máquina recibe la información que determina la longitud de los impulsos del motor desde una tabla de ocho bytes, denominada ANGULO. Al igual que en la versión para el BBC Micro, se utiliza un trozo de código adicional para cargar las posiciones ANGULO uniformemente desde una segunda tabla, NUEVAPOS, que es la tabla que se carga desde el programa controlador en BASIC.

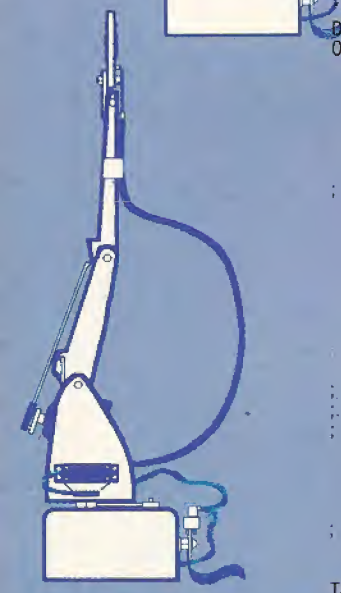
El programador de secuencias para el brazo utiliza el programa en código máquina que acabamos de describir para activar el brazo-robot. Similar a la versión para el BBC Micro, este programa permite controlar el brazo desde el teclado, guardando las posiciones clave, que se pueden luego reproducir o almacenar en cinta o disco. Entre esta versión y la versión para el BBC Micro existen, sin embargo, una o dos diferencias importantes. En primer lugar, al controlar el brazo-robot los motores tienen una tendencia a oscilar a través de una posición dada. Ello se debe a que el chip de video VIC-II del Commodore interfiere con la generación regular de interrupciones. La forma más simple de evitar este problema consiste en limpiar la pantalla cuando se emplea el programa cuña. Las subrutinas de las líneas 7000 y 8000 limpian la pantalla antes de entrar en la cuña y la restablecen al salir de la misma.

El segundo problema es que el BASIC Commodore hace que la comprobación de una gran cantidad de teclas (y las bifurcaciones correspondientes) sea un proceso lento. Por consiguiente, se utiliza la última subrutina en código máquina para comparar el código ASCII de una pulsación de tecla con una tabla de posibles teclas que tienen una función en la tabla. Cuando se encuentra una coincidencia en la tabla, se pasa como un número que puede utilizar el ON ... GOSUB del BASIC para llamar a la rutina adecuada.

Programador de secuencias para el brazo en el C64

Cargador en BASIC

```
10 REM ********************************
20 REM **                              **
30 REM **   CARGADOR DE BASIC PARA     **
40 REM **   CONTROLADOR BRAZO CBM      **
50 REM **                              **
60 REM ********************************
65 :
70 FOR I=49155 TO 49394
75 READ A:POKE I,A:CC=CC+A
80 NEXT I
90 READ CS:IF CS<> CC THEN PRINT"ERROR EN SUMA DE
   CONTROL":STOP
95 :
100 DATA169,255,141,3,221,169,62,141,1
110 DATA192,169,192,141,2,192,120,173
120 DATA20,3,174,1,192,141,1,192,142
130 DATA20,3,173,21,3,174,2,192,141,2
140 DATA192,142,21,3,169,16,133,251
150 DATA169,193,133,252,169,255,160,0
160 DATA145,251,136,208,251,88,96,8,72
170 DATA152,72,138,72,169,255,141,1
180 DATA221,162,7,169,255,24,106,72
190 DATA188,0,193,49,251,145,251,104
200 DATA202,16,243,160,46,136,208,253
210 DATA169,255,160,0,49,251,141,1,221
220 DATA200,208,248,162,7,169,255,188
230 DATA0,193,145,251,202,16,248,104
240 DATA170,104,168,104,40,108,1,192
250 DATA162,0,160,0,189,8,193,221,0
260 DATA193,240,14,176,6,222,0,193,76
270 DATA156,192,254,0,193,76,156,192
280 DATA200,232,224,4,208,228,32,169
290 DATA192,192,4,208,217,96,138,72
300 DATA152,72,160,255,174,0,192,136
310 DATA234,234,208,251,202,208,248
320 DATA104,168,104,170,96,0,255,255,0
330 DATA0,255,255,0,0,255,255,0,0,255
340 DATA255,0,0,72,138,72,152,72,32
350 DATA228,255,201,0,240,249,162,0
360 DATA221,191,192,240,7,232,224,17
370 DATA208,246,162,255,142,207,192
380 DATA104,168,104,170,104,96
390 DATA33097:REM"SUMA DE CONTROL"
```



Ian McKinnell

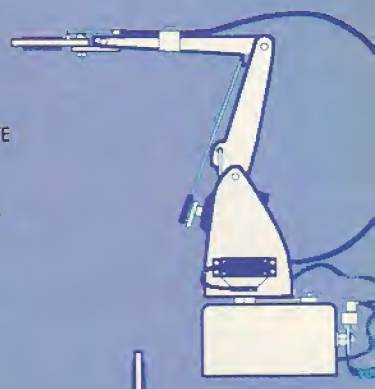
Listado assembly

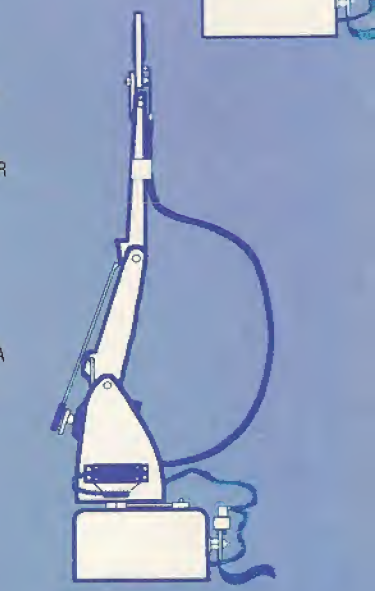
```
;.....
;.....
;.....
;..  CONTROLADOR BRAZO CBM  ..
;..                         ..
;.....
;.....
;
PUERTA=56577    ;REGISTRO DATOS PUERTA USUARIO
RDD=56579       ;REG DIRECCION DATOS PUERTA USUARIO
ANGULO=$C100    ;POSICION VALOR ANGULO
NUEVAPOS=$C108 ;TABLA POSICION
MOTTAB=$C110    ;TABLA REFERENCIA MOTOR
ZPAGE=$FB       ;PUNTERO PAGINA 0 A TABLA
IRQVEC=$0314    ;LOBYTE VECTOR IRQ

*=$C000
DELFAC *=*+1;ALMACENAMIENTO PARA FACTOR DEMORA
OURVEC *=*+2;ALMACENAMIENTO PARA NUESTRO VECTOR
        LDA   #$FF
        STA   DDR       ;ESTABLECER DDR EN SALIDA
        LDA   #<EVENT
        STA   OURVEC    ;APUNTA A EVENT
        LDA   #>EVENT   ;MANEJADOR
        STA   OURVEC+1
;
        SEI             ;APAGAR INTERRUPCIONES
        LDA   IRQVEC    ;INTERCAMBIAR VECTOR
        LDX   OURVEC    ;IRQ EXISTENTE POR
        STA   OURVEC    ;VECTOR NUESTRO
        STX   IRQVEC
        LDA   IRQVEC+1
        LDX   OURVEC+1
        STA   OURVEC+1
        STX   IRQVEC+1
;
;.... INICIALIZAR TABLA ....
;
        LDA   #<MOTTAB
        STA   ZPAGE
        LDA   #> MOTTAB
        STA   ZPAGE+1
;
        LDA   #$FF
        LDY   #$00
TABLA
        STA   (ZPAGE),Y
        DEY
        BNE   TABLA
        CLI             ;ENCENDER INTERRUPCIONES
;
;.... MANEJADOR EVENTOS ....
;
EVENT
        PHP
        PHA             ;GUARDAR REGISTROS
        TYA             ;EN PILA
        PHA
        TXA
        PHA

;  EMPEZAR IMPULSO, PARA ALGUNOS MOTORES SE
;PODRIA EMPEZAR ANTES DE RELLENAR LA TABLA
;Y ASI REDUCIR EL BUCLE DE ESPERA DE ABAJO..
;
        LDA   #$FF
        STA   PUERTA
;.. LLENAR TABLA CON EXCEPCIONES ..
        LDX   #$07
        LDA   #$FF
        CLC
EXCEPT
        ROR   A
        PHA             ;PATRON BITS
        LDY   ANGULO,X  ;TOMAR DESPL. MOTOR X
        AND   (ZPAGE),Y ;CONSERVAR PATRON EXISTENTE
        STA   (ZPAGE),Y ;PERO MODIFICADO PARA MOTOR X
        PLA
        DEX
        BPL   EXCEPT
;.. AHORA LA TABLA ESTA CARGADA ..
        LDY   #$30
WAIT
        DEY             ;ESPERAR
        BNE   WAIT      ;UN POCO

        LDA   #$FF      ;TODOS IMPULSOS ENCENDIDOS
        LDY   #$00
LOOP
        AND   (ZPAGE),Y ;PERO DESENMASCARA CADA ELEMENTO
        STA   PUERTA    ;DE LA TABLA POR TURNO
        INY
        BNE   LOOP
;
        LDX   #$07
        LDA   #$FF
CLEAR
        LDY   ANGULO,X  ;BORRAR TODAS EXCEPCIONES
        STA   (ZPAGE),Y
        DEX
        BPL   CLEAR
;.. TODOS LOS IMPULSOS DEBEN HABER TERMINADO AHORA ..
        PLA
        TAX
        PLA             ;RESTAURAR REGISTROS
        TAY
        PLA
        PLP
        JMP   (OURVEC)  ;SALTAR A RUTINA
                        ;IRQ NORMAL
;
;.... MOVER EL MOTOR UNIFORMEMENTE ....
;
LOOP2
        LDX   #$00
        LDY   #$00
LOOP1
        LDA   NUEVAPOS,X;NUEVA POSICION PARA SERVO X
        CMP   ANGULO,X  ;ES LA MISMA QUE LA POS. ANTIGUA?
        BEQ   MOVED     ;SI=NO HACER NADA
        BCS   ADD       ;SI>SUMAR
        DEC   ANGULO,X  ;SINO RESTAR
        JMP   INCRX
ADD
        INC   ANGULO,X
        JMP   INCRX
MOVED
        INY
INCRX
        INX
        CPX   #$04      ;HECHOS LOS 4 SERVOS
        BNE   LOOP1     ;SI NO SIGUIENTE SERVO
        JSR   WAITER    ;BREVE PAUSA
        CPY   #$04      ;SI Y=4 ENTONCES TODOS MOVIDOS
        BNE   LOOP2
        RTS
;
WAITER
        TXA
        PHA             ;GUARDAR REGS X & Y
        TYA
        PHA
        LDY   #$FF
        LDX   DELFAC    ;TOMAR FACTOR DEMORA
ENCORE
        DEY
        NOP             ;256*8 CICLOS RELOJ
        NOP             ;ESPERA
        BNE   ENCORE
        DEX             ;SI MAS DEMORA ENTONCES
        BNE   ENCORE    ;REPETIR
        PLA
        TAY
        PLA
        TAX
        RTS
;.... COMPROBAR TECLADO ....
;
KEYTAB *=*+16
RESULTADO *=*+1
GETIN=$FFE4
;
CHECK
        PHA
        TXA
        PHA
        TYA
        PHA
NOKEY
        JSR   GETIN     ;TOMAR UN CARACTER
        CMP   #$00      ;NINGUN CAR?
        BEQ   NOKEY
        LDX   #$00
COMPAR
        CMP   KEYTAB,X
        BEQ   EXIT      ;ESTA EL CAR. EN LA TABLA?
        INX
        CPX   #17
        BNE   COMPAR
        LDX   #$FF      ;NO HALLADA SEÑAL
EXIT
        STX   RESULTADO ;ALMACENAR DESPL. KEYTAB
        PLA
        TAY
        PLA
```

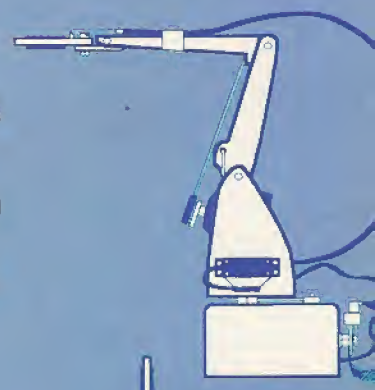
```
      TAX
      PLA
      RTS
```

Programador de secuencias para el brazo-robot

```
 10 REM  ************************
 20 REM * ************************
 30 REM **                       **
 40 REM **  PROGRAMADOR CBM      **
 50 REM **   SECUENCIAS BRAZO    **
 70 REM **                       **
 80 REM  /************************
 90 REM  ************************
 95 :
 96 DN=8:REM PARA CASSETTE DN=1
 97 IF A=0 THEN A=1:LOAD"ROBARM.HEX",DN,1
100 GOSUB 1000:REM PREPARACION
110 PRINT CL$
120 X=5:Y=6:GOSUB 10000:PRINT"TE GUSTARIA:  "
130 X=4:Y=10:GOSUB 10000:PRINT"1....PROGRAMAR NUEVA
    SECUENCIA BRAZO"
140 X=4:Y=12:GOSUB 10000:PRINT"2....AGREGAR
    MOVIMIENTOS A UN PROGRAMA"
150 X=4:Y=14:GOSUB 10000:PRINT"3....REPRODUCIR UN
    ARCHIVO"
160 X=4:Y=16:GOSUB 10000:PRINT"4....SALIR DEL
    PROGRAMA"
170 GET G$:IF G$="" THEN 170
180 IF G$="1" THEN C=0:LM=0:REM RESTABLECER PUNTEROS
    MATRIZ
190 IF G$="1" OR G$="2" THEN GOSUB 2000:GOSUB
    3000:GOSUB 4000:GOSUB 5000
200 IF G$="3" THEN GOSUB 6000:GOSUB 4000
210 IF G$="4" THEN PRINT CL$:END
220 GOTO 110
500 REM **** FUNCIONES ****
540 DX=DX+1:RETURN
550 DX=DX-1:IF DX<0 THEN DX=0
555 RETURN
560 P=PEEK(NP)+DX:IF P<256 THEN POKENP,P
565 RETURN
570 P=PEEK(NP)-DX:IF P>0 THEN POKENP,P
575 RETURN
580 P=PEEK(NP+1)+DX:IF P<256 THEN POKENP+1,P
585 RETURN
590 P=PEEK(NP+1)-DX:IF P>0 THEN POKENP+1,P
595 RETURN
600 P=PEEK(NP+2)-DX:IF P>0 THEN POKENP+2,P
605 RETURN
610 P=PEEK(NP+2)+DX:IF P<256 THEN POKENP+2,P
615 RETURN
620 P=PEEK(NP+3)+3*DX:IF P<256 THEN POKENP+3,P
625 RETURN
630 P=PEEK(NP+3)-3*DX:IF P<256 THEN POKENP+3,P
635 RETURN
640 FOR I=0 TO 3:POKE NP+I,R%(C,I):NEXT I:RETURN
650 C=C+1:IF C>MC THEN C=0
655 RETURN
660 C=C-1:IF C<0 THEN C=MC
665 RETURN
670 FOR I=0 TO 3:R%(C,I)=PEEK(NP+I):NEXT:C=C+1:RETURN
680 FOR I=0 TO 3:R%(C,I)=PEEK(NP+I):NEXT:C=C+1:RETURN
999 :
1000 REM **** PREPARACION ****
1010 C=0:NS=3:REM N.DE SERVOS-1
1020 LM=0:OC=0:DF=10:REM FACTOR DE DEMORA
1030 MC=100:REM MAX CONTADOR
1040 DIM R%(MC,NS):REM MATRIZ POSICIONES TECLAS
1050 CL$=CHR$(147):REM LIMPIAR PANTALLA
1060 DW$="":FOR I=1 TO 25:DW$=DW$+CHR$(17):NEXT I:REM
     CURSOR ABAJO
1065 POKE 650,128:REM EST. TECLAS PARA REPETIR
1070 DL=49152:NP=49416:REM DIRECCIONES DEMORA Y
     POSICION
1075 GS=49155:OFF=49170:REM DIRECCIONES SISTEMA CUÑA
     ON/OFF
1077 MOVER=49281:REM MOVER DIRECCIONES SYS SERVOS
1078 KEYTAB=49343:RESULTADO=49359:REM DIRECCIONES
     PULSACION TECLA
1079 CHECK=49360:REM DIRECCION SYS COMPROBACION TECLA
1080 REM ** LEER DATOS ASC PULSACION TECLA **
1090 FOR I=0 TO 14:READ A:POKE KEYTAB+I,A:NEXT I
1100 DATA 73,68,157,29,145,17,65,90,88,67
1110 DATA 82,78,66,83,81
1200 REM ** PONER A CERO POSICIONES NUEVAPOS **
1210 FOR I=0 TO 3:POKENP+I,0:NEXT I
1990 RETURN
1999 :
2000 REM **** INFORME ****
2010 PRINT CL$
2020 X=5:Y=5:GOSUB 10000:PRINT"POR FAVOR UTILIZA :"
2030 X=1:Y=7:GOSUB 10000:PRINT"TECLAS CURSOR....PARA
     D/I Y 1ER. BRAZO AR/AB"
2040 X=1:Y=9:GOSUB 10000:PRINT"A & Z..... PARA
     2DO.BRAZO AR/AB"
2050 X=1:Y=11:GOSUB 10000:PRINT"X & C..... PARA
     ABRIR/CERRAR PINZA"
2060 X=1:Y=13:GOSUB 10000:PRINT"S..... PARA
     SALVAR UNA POSICION"
2070 X=1:Y=15:GOSUB 10000:PRINT"Q..... PARA
     VOLVER AL MENU"
2080 X=1:Y=17:GOSUB 10000:PRINT"R..... PARA
     MOVER A POSICION SALVADA"
2090 X=1:Y=19:GOSUB 10000:PRINT"N & B..... SIGUIENTE
     Y PREVIO CONTADOR"
2100 X=1:Y=21:GOSUB 10000:PRINT"E.....
     ESTABLECER NUEVO CONTADOR"
2110 X=1:Y=23:GOSUB 10000:PRINT"I & D..... PARA INC.
     Y DEC. VELOCIDAD"
2120 X=2:Y=2:GOSUB 10000:PRINT"CONTADOR=";C;"  ";
2130 FOR I=0 TO 3:PRINT R%(C,I);"  ";:NEXT:PRINT
2135 GET A$:IF A$="" THEN 2135
2140 RETURN
2999 :
3000 REM **** PROGRAMAR BRAZO ****
3005 GOSUB 7000:REM BORRAR PANTALLA Y EMPEZAR CUÑA
3010 POKE DL,1:REM ESTABLECER FACTOR DEMORA EN 1
3020 DX=8:REM CAMBIAR VELOCIDAD
3030 SYS CHECK:REM COMPROBAR PULSACION TECLA
3035 R=PEEK(RESULTADO)+1:IF R>15 THEN 3030:REM TECLA
     NO VALIDA
3037 ON R GOSUB 540,550,560,570,580,590,600,610,620,
     630,640,650,660,670
3190 IF C>LM THEN LM=C-1:REM REGISTRAR MAX CONTADOR
     HASTA AHORA
3220 OC=C
3260 SYS MOVER:REM MOVER SERVOS C/M
3270 IF R=15 OR C>MC THEN GOSUB 8000:RETURN:REM
     APAGAR CUÑA Y SALIR
3280 GOTO 3030:REM REPETIR
3999 :
4000 REM **** REPRODUCIR SECUENCIA MOVIMIENTOS ****
4010 PRINT CL$
4020 X=5:Y=23:GOSUB 10000:PRINT"REPRODUCIR SECUENCIA
     (S/N),R REPITE"
4030 GET AN$:IF AN$<>"S" AND AN$<>"N" AND AN$<>"R"
     THEN 4030
4040 IF AN$="N"THEN RETURN
4050 IF AN$<>"R" THEN X=5:Y=22:GOSUB
     10000:INPUT"FACTOR DE DEMORA 1-255":DF
4060 IF DF<0 OR DF>255 THEN 4050:REM COMPROBAR
     ESCALA
4070 POKE DL,DF:REM ESTABLECER REGISTRO FACTOR
     DEMORA
4075 GOSUB 7000:REM BORRAR PANTALLA Y COMENZAR
     CUÑA
4080 FOR I=0 TO LM
4090 X=5:Y=2:GOSUB 10000:PRINT"N. EN SECUENCIA=  ";I;
     "  ";
4100 FOR S=0 TO 3
4110 POKE NP+S,R%(I,S)
4120 NEXT S:SYS MOVER:REM MOVE SERVOS C/M
4130 NEXT I
4135 GOSUB 8000:REM RESTAURAR PANTALLA Y APAGAR CUÑA
4140 GOTO 4010:REM REPETIR
4999 :
5000 REM **** GUARDAR UNA SECUENCIA ****
5010 PRINT CL$
5020 X=5:Y=10:GOSUB 10000:PRINT"GUARDAR SECUENCIA EN
     ARCHIVO (S/N)"
5030 GET AN$:IF AN$<>"S" AND AN$<>"N" THEN 5030
5040 IF AN$="N" THEN RETURN
5050 X=5:Y=12:GOSUB 10000:INPUT"NOMBRE ARCHIVO";FL$
5060 OPEN 3,DN,3,"00:"+FL$+",DATA,S,W":REM ABRIR
     ARCHIVO
5070 PRINT #3,LM
5080 FOR A=0 TO LM:FOR B=0 TO 3
5090 PRINT #3,R%(A,B):NEXT B,A
5100 PRINT #3:CLOSE3
5120 RETURN
5999 :
6000 REM **** CARGAR UN ARCHIVO ****
6010 PRINTCL$
6020 X=5:Y=10:GOSUB 10000:INPUT"NOMBRE ARCHIVO A
     CARGAR":FL$
6040 OPEN 3,DN,3,"00:"+FL$+",DATA,S,R":REM ABRIR
     ARCHIVO
6050 INPUT #3,C:LM=C
6060 FOR A=0 TO C:FOR B=0 TO 3
6070 INPUT #3,R%(A,B):NEXT B,A
6080 PRINT #3:CLOSE3
6100 RETURN
6999 :
7000 REM **** COMENZAR CUÑA ****
7010 POKE 53265,PEEK(53265)AND 239:REM BORRAR
     PANTALLA
7015 SYS GS
7020 RETURN
7999 :
8000 REM **** DETENER CUÑA ****
8010 POKE 53265,PEEK(53265)OR 16:REM BORRAR PANTALLA
8015 SYS OFF
8020 RETURN
8999 :
10000 REM **** POSICIONAR CURSOR EN X,Y ****
10010 PRINTCHR$(19);:PRINTTAB(X);LEFT$(DW$,Y);
10020 RETURN
```







Trabaja la vista

Vamos primero a examinar los gráficos a colores y la organización de la memoria que necesita el chip VIC-II, para obtener después inusitados efectos visuales

El Commodore 64 tiene ocho modos básicos de gráficos en pantalla. En los gráficos de baja resolución el carácter puede albergarse en la ROM o en la RAM y visualizarse en uno de estos tres modos: estándar, multicolor o ampliado *(extended)*. Se distinguen además dos modos en alta resolución: estándar y multicolor. Aparte de estos modos básicos, se dispone de diversas variantes: se puede hacer que la pantalla cuente con 38 columnas, en vez de las 40 columnas habituales y/o 24 filas, en vez de 25. Tales características suelen usarse junto con el *scroll* (desplazamiento) lento vertical u horizontal. El *scroll* en el Commodore 64 se controla mejor desde código máquina, ya que los datos a visualizar en desplazamiento deben pasar bastante rápidamente por la RAM de pantalla. Ya diseñamos anteriormente un programa para desplazamientos suaves horizontales de pantalla empleando el modo de pantalla de 38 columnas.

Si una pantalla en alta resolución es empleada con un programa largo, puede que la memoria se resienta. El Commodore 64 facilita una gran libertad en el posicionamiento de la pantalla en memoria tanto en alta como en baja resolución, por lo que comenzaremos con un breve análisis de cómo puede el programador trasladar ambas pantallas por la memoria. Veremos en el siguiente capítulo que si se emplea el código máquina para hacer los puntos, incluso se puede colocar la pantalla en alta resolución *detrás* de la ROM del intérprete del BASIC. Bien pensado, se trata de un excelente recurso dado que si sólo se está ejecutando código máquina la ROM del intérprete ocupa ocho Kbytes de una memoria valiosa inútilmente. Una interesante coincidencia nos dice que son justamente ocho Kbytes los que se necesitan para una pantalla de alta resolución.

Esta función del Chip para Interface de Video 6566/67 (VIC-II) para generar datos de visualización del video es la que se pasa al televisor o pantalla. Para ello el VIC-II debe saber leer datos desde la RAM o la ROM. Merece la pena estudiar el modo como el VIC-II obtiene sus datos, ya que esto es la base del comportamiento más recóndito del Commodore 64.

Cuadro 1

Modelo de bits del par de pixels		Color	Determinado por
0	0	Fondo de pantalla	53281 ($D021) Bits 0 al 4
0	1	Multicolor #1	53282 ($D022) Bits 0 al 4
1	0	Multicolor #2	53283 ($D023) Bits 0 al 4
1	1	Color primer plano	Bits 0 al 3 del cuarteto color

Cuadro 2

Código pantalla	Bit 7	Bit 6	Registro color fondo
64 — 127	0	1	53282 ($D022) Color 1 ampliado
128 — 191	1	0	53283 ($D023) Color 2 ampliado
192 — 255	1	1	53284 ($D024) Color 3 ampliado

Modo multicolor

Ya quedaron descritos tanto la visualización habitual en baja resolución como el modo en mapa de bits en el Commodore 64. Vamos a examinar ahora otras dos maneras de emplear los gráficos de la máquina.

Contrastando con dos de ellos, podemos obtener, en el modo multicolor, cuatro colores dentro de la celda individual de un carácter. Pero en esta resolución horizontal perdemos en otro aspecto, y es que ahora ésta trabaja por pares de pixels y no pixel a pixel. El modo multicolor puede emplearse con gráficos tanto en alta como en baja resolución, aunque los puntos coloreados se obtienen de un modo algo distinto en el modo de alta resolución. El siguiente POKE activa y desactiva desde el BASIC dicho modo multicolor.

```
POKE 53270,PEEK (53270)OR16
POKE 53270,PEEK (53270)AND239
```

En baja resolución, si el modo multicolor está activo y el bit 4 del cuarteto o semibyte asociado del color está puesto a uno, el carácter se interpreta en modo multicolor, donde los tres bits inferiores especifican el color. Esto quiere decir que los caracteres que tienen cuartetos asociados de color con valor entre 0 y 7 son interpretados normalmente; mientras que si el código del color está entre 8 y 15, el carácter se visualizará en modo multicolor. En el cuadro 1 mostramos cómo se determinan los colores de cada par de pixels.

Con el contenido de las direcciones 53282 y 53283, podemos cambiar instantáneamente el color de cada par de pixels asociado en Multicolor. Es de resaltar que el Multicolor trabaja mejor con caracteres definidos por el usuario, es decir, las agrupaciones de bits tienen en cuenta el hecho de ser interpretados como pares.

Otro modo de gráficos disponible para un programador del Commodore 64 es el modo de color ampliado *(extended)*. Este modo permite controlar el color de fondo de los primeros 64 caracteres dentro de la matriz de caracteres. El modo de color ampliado no puede emplearse junto con el modo multicolor. Las siguientes líneas de BASIC activan y desactivan el modo de color ampliado respectivamente:

```
POKE 53265,PEEK(53265)OR64
POKE 53265,PEEK(53265)AND191
```

Un mismo carácter puede aparecer en la pantalla hasta con cuatro colores de fondo diferentes, uno de los cuales es el color de fondo de la pantalla. Los caracteres restantes no pueden visualizarse en el modo de color ampliado, ya que los bits 6 y 7 del código de pantalla se emplean para controlar indirectamente el color del carácter. Por ejemplo, el código de pantalla para "A" es 1, y el de "A" en un campo invertido es 65. Pero si introducimos (POKE) 65 en la pantalla en modo de color ampliado, no obtendremos una "A" en campo invertido sino una "A" habitual con un color de fondo determinado por el contenido de la dirección 53282 ($D022). Igualmente, introduciendo (POKE) 129 en la pantalla obtendremos una "A" normal, pero con un color de fondo dictado por el contenido de la dirección 53283 ($D023). El cuadro 2 muestra la relación que existe entre los códigos de pantalla y los registros de color.

Gestión de la memoria

El chip VIC-II ve desde el 6510 un mapa distinto y mucho más simple de memoria. Simultáneamente el VIC-II sólo puede ver uno de los cuatro bloques de 16 Kbytes (bancos) de memoria. Podemos imaginarnos este banco de 16 Kbytes como la "ventana" del VIC-II sobre la memoria. La dirección de base de esta ventana puede tomar uno de los cuatro valores que se encuentran bajo el control del software

```
1000 REM ** SELECCION BANCO/VENTANA DEL
     VIC **
1010 POKE 56578,PEEK(56578)OR3:REM BITS 0,1
     DE CIA#2 DE RDD,PARA SALIDA
1020 VT=3:REM SELECCIONA LA VENTANA
     NORMAL
1030 POKE 56576,PEEK(56576)AND252)ORVT:REM
     BITS 0,1 DE CIA#1 PUERTA A
```

VT puede tomar aquí valores del 0 al 3. Cada vez encontraremos el valor de VT en curso como PEEK (56576)AND3. La dirección correspondiente a la parte inferior de la ventana de 16 Kbytes sobre la memoria se calcula por medio de la fórmula: VB=16384*(3−VT), en la que el valor VT proporciona las direcciones correspondientes:

VT	Dirección inicio ventana
0	49152 ($C000)
1	32768 ($8000)
2	16384 ($4000)
3	0 ($0000)

Dentro de la ventana del VIC-II, el 6510 espera ver la memoria de pantalla y una imagen de la ROM de caracteres en baja resolución (o datos en alta resolución, si se ha seleccionado el modo de alta resolución). Puede que también necesite encontrar datos de sprites, y en ese caso los ha de ver dentro de la ventana.

Los punteros de cada uno de los ocho sprites se encuentran colocados al final de la memoria de pantalla (y con la pantalla deberán moverse).

El desplazamiento del inicio de la memoria de pantalla desde la base de la actual ventana del VIC-II se controla por medio de los cuatro bits superiores del registro de control del VIC-II en la dirección 53272 ($D018). Empleando estos cuatro bits, podemos colocar la pantalla en cualquiera de los bloques de 16 Kbytes que integran la ventana.

```
1040 REM ** SELECCION DESPLAZAMIENTO
     PANTALLA DE LA BASE DE LA VENTANA **
1050 SD=1:REM SELECCIONA EL
     DESPLAZAMIENTO NORMAL
1060 POKE 53272,(PEEK(53272)AND15)OR
     16*SD
```

SD puede tomar valores entre 0 y 15. Siempre encontraremos el valor en curso de SD empleando PEEK(53272)AND15. La dirección correspondiente a la base de la pantalla actual en la memoria se calcula o bien así PT=VB+1024*SD (base ventana más desplazamiento) o bien así:

```
PT=16384*(3−(PEEK(56576)AND3))+64*(PEEK
(53272)AND240)
```

El problema a la hora de mover la pantalla de un sitio a otro radica en que la RAM del color no se mueve. Por ello, si deseamos obtener una pantalla alternativa, habremos de emplear una pequeña rutina en código máquina para intercambiar la memoria del color dentro o fuera de un buffer o memoria intermedia según convenga. Es lo que se hizo como base de un programa para pantallas alternativas en el Commodore 64.

Otro factor a considerar es que si se desea imprimir (PRINT) en la nueva pantalla, es necesario avisar al *sistema operativo* (y no al chip de video) sobre dónde se encuentra la nueva pantalla. Esto se realiza con un POKE (o STA) en la dirección 648 ($0288), que es un puntero que contiene el byte superior de la dirección de base de la memoria de pantalla. Si calculamos PT como se ha hecho más arriba, este trabajo lo hará la siguiente instrucción en BASIC:

```
POKE 648,INT(PT/256)
```

Para seleccionar la dirección de base de la matriz de caracteres o pantalla en alta resolución, emplearemos el siguiente código:

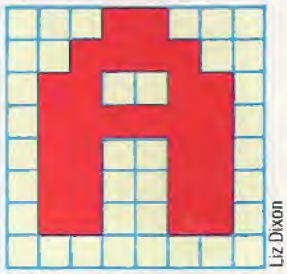
```
1070 REM ** SELECCION DEL DESPL. ALT. RES.
     /MAT. CAR DESDE BASE VENTANA **
1080 AD=4:REM SELECCIONA DESPLAZAMIENTO
     NORMAL
1090 POKE 53272,(PEEK(53272)AND240)
     OR2^AD
```

En principio, AD puede tomar valores entre 0 y 7, pero en la práctica existen limitaciones. Con VT, que vale 1 o 3, no podemos dar a AD el valor 2 o el 3, ya que éste es donde el VIC-II ve la imagen normal del carácter de ROM. Igualmente, un valor grande en AD pondrá la parte superior de la memoria en alta resolución fuera del alcance del VIC-II. La dirección correspondiente de la base de la actual pantalla en alta resolución o de matriz de caracteres se calcula así MC=VB+2048*AD (base ventana más desplazamiento), o bien así:

```
MC=16384*(3−(PEEK(56576)AND3))+1024*
(PEEK(53272)AND14)
```

Activando estos registros podemos mover toda la memoria de visualización en video de un sitio a otro siempre que lo requiera el programa.

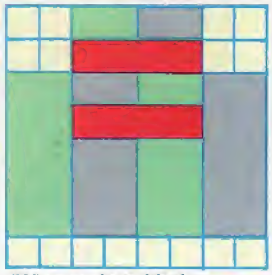
Colores coordinados
En el modo de visualización normal del Commodore 64, todo bit puesto a uno en los ocho bytes que definen un carácter se visualiza según el color actual de primer plano. Los bits con valor 0, por el contrario, se visualizan con el color del fondo de la pantalla. Al seleccionar el modo multicolor, los modelos de bits dejan de ser interpretados como bits individuales para serlo por pares. Las 4 combinaciones posibles de un par de bits representan los colores de primer plano y de fondo así como dos multicolores extra



Liz Dixon

"A" en modo de visual. normal

```
0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0
```



"A" en modo multicolor

```
0 0   0 1   1 0   0 0
0 0   1 1   1 1   0 0
0 1   1 0   0 1   1 0
0 1   1 1   1 1   1 0
0 1   1 0   0 1   1 0
0 1   1 0   0 1   1 0
0 1   1 0   0 1   1 0
0 0   0 0   0 0   0 0
```

Clave

01 MULTICOLOR 1

10 MULTICOLOR 2

11 COLOR DE PRIMER PLANO

00 COLOR DE FONDO

Al salir la luna

En "Knight Lore" (El caballero Lore), juego que cuenta con unos gráficos tridimensionales excelentes, el héroe se transforma en hombre lobo

La habitación del brujo

La bota

La copa

La guarida del licántropo El objetivo de *Knight Lore* consiste en llevar a la caldera el objeto necesario para impedir que el caballero se convierta en un hombre lobo. Para hacer esto, primero es preciso visitar al brujo para descubrir qué necesitará. Entonces puede iniciarse la búsqueda en serio. Aunque los objetos pueden cambiar de posición, siempre se los encontrará en las mismas habitaciones. Por ejemplo, para llegar a la bota uno debe subir las mesas a la columna y trepar. Por otra parte, para llegar a la copa hay que esperar a que se abra la puerta correcta

Ian McKinnell

Ultimate siempre se ha destacado por producir juegos de calidad para el Sinclair Spectrum. La empresa ha creado una larga lista de juegos de laberinto en los cuales el jugador debe luchar para abrirse paso a través de las extrañas y maravillosas criaturas que habitan las diversas habitaciones. Mientras tanto, el jugador también debe recoger tesoros, así como talismanes y otras bagatelas místicas que, una vez unidas, completarán el juego. Ejemplos de este tipo son *Atic atac* y *Sabre Wulf*.

A pesar de la popularidad de estos juegos entre los usuarios del Spectrum, ahora la empresa se ha desmarcado de la aventura recreativa bidimensional tipificada por *Sabre Wulf*. Aunque guarda algunas semejanzas con las anteriores creaciones de Ultimate, *Knight Lore* es un tipo de juego muy diferente. El argumento es el siguiente: un caballero (en realidad, Sabre Man, el héroe de varias aventuras anteriores) busca el castillo de un hechicero para descubrir el conjuro que impedirá que él se convierta para siempre en un hombre lobo. El jugador dispone de cuarenta días con sus respectivas noches para hallar la respuesta, cuyo transcurso se va visualizando en la parte inferior de la pantalla; en la esquina inferior derecha hay una ventana que contiene un "cuadrante solar". Durante el día el jugador es Sabre Man. Sin embargo, cuando en la ventana aparece la luna, se transforma, con espasmos en el mejor estilo de Lon Chaney, en el Hombre Lobo. Afortunadamente, ello no tiene ninguna influencia en la capacidad del jugador para desplazarse, a excepción del momento de la transformación propiamente dicho, lo que puede ser un inconveniente si está intentando librarse de algún personaje monstruoso.

Al comenzar el juego, lo primero que usted percibirá es la incorporación de gráficos tridimensionales a la aventura. Estos gráficos son los mejores que se han visto hasta ahora para el Spectrum, incluyendo *Ant attack*. No obstante, a diferencia de este último, en *Knight Lore* no se puede cambiar el ángulo de visión, pero esto se compensa gracias a las ingeniosas formas en que se han construido las diversas habitaciones.

Knight Lore no es una aventura recreativa *per se*, sino más bien una serie de acertijos lógicos que se han de resolver. En este sentido, se acerca al auténtico juego de aventuras, al plantear arduos problemas que deben resolverse para poder seguir avanzando. Aunque muchas de las habitaciones están vacías, otras le ofrecerán desafíos que deberá resolver para salir por el otro extremo de la habitación. Una de ellas, por ejemplo, está habitada por pequeñas criaturas con apariencia de fantasmas que corren por el suelo: si consiguen tocar a Sabre Man, pierde una vida.

Los enigmas presentan diversos niveles de dificultad. Algunos son comparativamente fáciles, otros son de una complejidad diabólica, y varios sólo constituyen pistas falsas. Usted puede, por ejemplo, llegar a una habitación con bloques, encima de los cuales hay grandes globos con púas. Si intenta cruzar los bloques, estas esferas punzantes simplemente pincharán su entusiasmo; la respuesta no es otra que ¡pasar caminando por entre los bloques! Otro truco que utilizan los programadores es valerse de una representación bidimensional de un espacio tridimensional. Con frecuencia, bloques que parecen estar sobre el suelo en realidad están flotando en el aire, y si Sabre Man cae debajo de uno de ellos, no tendrá forma alguna de salir como no sea perdiendo una vida entre las púas e intentando volver a cruzar la habitación.

En muchas de las habitaciones hay objetos a recoger, tales como piedras preciosas, cálices y pociones. Una vez se llega a éstos, para cogerlos hay que saltar encima del objeto y tirar de la palanca de mando hacia atrás. Con ello el objeto quedará incluido en el "inventario" del jugador, que se visualiza en la esquina inferior izquierda de la pantalla. Esta maniobra puede ser útil para ganar altura cuando hay que saltar una pared que, de otra forma, sería insuperable.

Sabre Man puede moverse en las cuatro direcciones, utilizando ya sea el teclado o bien la palanca de mando. Los saltos se consiguen pulsando las teclas de la tercera fila o bien el botón de disparo. La orientación correcta de Sabre Man es esencial para desarrollar con éxito el juego; saltar en la dirección equivocada, por ejemplo, puede ser fatal. Quizá usted no tenga más remedio que pasar los primeros juegos simplemente familiarizándose con los mandos, para obtener un "ajuste exacto" de la orientación correcta, dado que movimientos muy ligeros de la palanca de mando pueden alterar la forma en que quede orientado Sabre Man.

Knight Lore es un juego fascinante y responde al elevado estándar de Ultimate. Incluso los jugadores a los que no les agrade encontrarse de vez en cuando con enigmas el juego les resultará sumamente absorbente, porque las soluciones a muchos de los problemas exigen respuestas rápidas con las teclas o el botón de disparo, además de exigir una mente aguda y atenta.

Knight Lore: Para el Spectrum de 48 K
Editado por: Ashby Ltd, Ashby de la Zouch, Leicestershire, LE6 5JU, Gran Bretaña
Autores: Ultimate Play the Game
Palanca de mando: Opcional
Formato: Cassette

mi COMPUTER